TVF-AD: TEMPORAL-VISUAL FUSION FOR TIME-SERIES ANOMALY DETECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Detecting abnormal temporal dependencies is fundamental to time series anomaly detection (TSAD), yet accurately modeling temporal dependencies, especially in multivariate time series, remains an open challenge. Recent approaches leverage pretrained visual models to capture temporal structures, showing promising performance by transferring structured visual semantics to sequential data and enhancing temporal structure modeling. However, such methods often rely heavily on visual priors and may fail to capture fine-grained temporal dynamics and domain-specific dependencies inherent in time series. To address this limitation, we propose **TVF-AD**, a novel hybrid temporal–visual representation-based framework for TSAD that integrates general visual priors with specialized temporal representations. Our framework effectively bridges cross-modal structural knowledge and sequential dynamics, enabling more accurate and robust anomaly detection, while maintaining competitive efficiency by optimizing only lightweight temporal encoding and fusion modules. Extensive experiments on benchmark datasets demonstrate that our method consistently outperforms state-of-the-art approaches.

1 INTRODUCTION

Multivariate time-series anomaly detection aims to identify abnormal patterns that deviate from normal system behaviors in multi-sensor sequential data. It is an important problem in a wide range of applications, including industrial monitoring (Zhang et al., 2019), cloud service reliability (Ren et al., 2019), and energy system management (Zhang et al., 2021). Despite extensive studies, this task remains challenging due to complex temporal dependencies, diverse anomaly patterns, and the scarcity of anomaly annotations in real-world scenarios (Su et al., 2019; Deng & Hooi, 2021; Xu et al., 2018). These factors make it difficult to learn discriminative representations for abnormal behaviors, especially when anomalies are subtle, heterogeneous, and weakly distinguishable from normal patterns.

Recently, representation learning has emerged as an important direction in TSAD (Darban et al., 2025; Park & Kang, 2026). However, most existing methods still model windows or patches mainly in the temporal domain, which is effective for capturing local dynamics and point-wise deviations but may be less sufficient for anomalies reflected in structural, periodic, or frequency-related patterns. Meanwhile, other studies have explored images or 2D representations of time series, including recurrence plots (Eckmann et al., 1995), Gramian Angular Fields (Wang & Oates, 2015a), Markov Transition Fields (Wang & Oates, 2015b), grayscale-image (Chen et al., 2025) and 2D temporal variation modeling approaches such as TimesNet (Wu et al., 2023).

As illustrated in Fig. 1, which shows 2D t-SNE visualizations of the temporal and visual representations, we observe clear complementarity between temporal representations (from raw sequence) and visual representations (from CWT-based time–frequency images) across different anomaly types. In Fig. 1(a), the anomaly is created by replacing a normal subsequence with another subsequence from a different position in the same series. The resulting segment remains temporally smooth, making it hard for temporal representations learned from raw windows or patches to identify, but its structural pattern becomes inconsistent with the surrounding context, whereas the visual representation can provide better separability. In Fig. 1(b), the anomaly corresponds to a noticeable frequency change, to which the visual representation is also more sensitive because such differences become more explicit in the time–frequency view. By contrast, the anomalies in Fig. 1(c) and Fig. 1(d) are domi-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

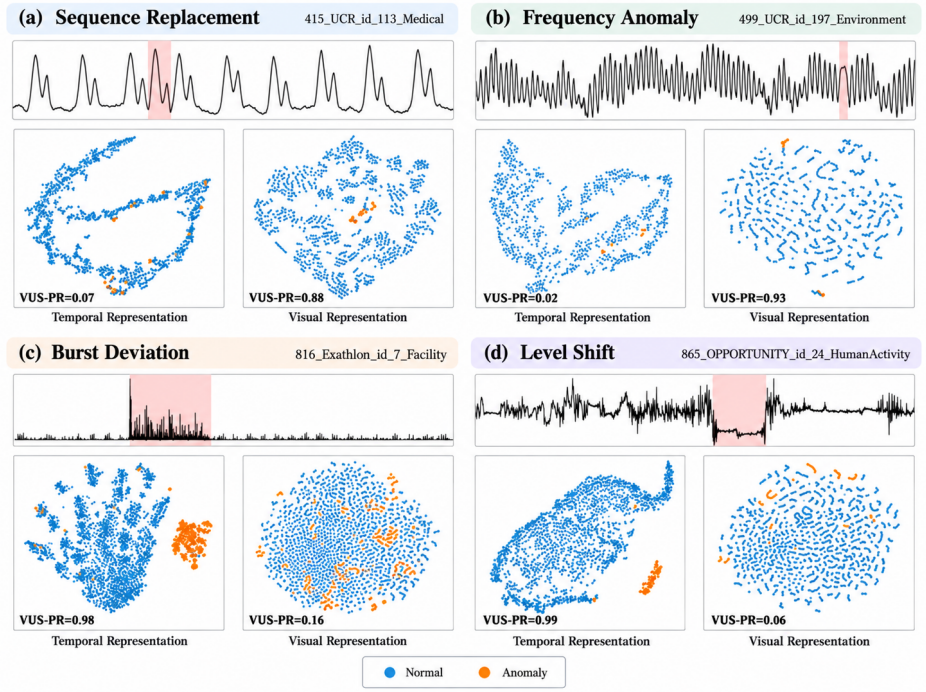


Figure 1: Temporal–Visual complementarity for different anomaly types. (a) Sequence replacement with contextual inconsistency. (b) Frequency anomaly with changed oscillation pattern. (c) Burst deviation with increased local amplitude or variance. (d) Level shift over a contiguous interval.

nated by local temporal deviations: Fig. 1(c) shows a burst segment with increased local amplitude or variance but no obvious frequency shift, while Fig. 1(d) reflects a local level/state change over a contiguous interval. For these cases, temporal representations preserve fine-grained sequential dynamics and local value transitions more directly, yielding clearer anomaly separation than the visual representation.

Motivated by the above complementarity, we propose **TVF-AD**, a compact temporal–visual fusion framework for representation-based TSAD. For each input window, TVF-AD uses a lightweight temporal encoder to model the raw multivariate sequence and a frozen pretrained visual encoder to extract complementary features from its CWT-based time–frequency image. The two views are then adaptively fused to learn a compact cross-view representation that preserves fine-grained temporal dynamics while incorporating structural and frequency-related visual cues. To keep the framework efficient, visual features are extracted offline rather than through end-to-end visual training, and the fused representations are used for anomaly discrimination and memory-bank-based scoring at inference. In this way, TVF-AD exploits visual priors without sacrificing the compactness and efficiency required in practical TSAD systems.

The main contributions of this paper are summarized as follows:

- We propose **TVF-AD**, a compact temporal–visual fusion representation-based framework for TSAD, which combines lightweight temporal modeling with offline visual features from a frozen pretrained visual encoder to efficiently exploit visual priors for anomaly detection.
- We show that complementary visual structural information can help capture anomaly cues that are difficult to identify from raw temporal values alone, and provide empirical analysis on key design choices of time-series-to-image conversion, especially variable grouping.
- Extensive experiments on the **TSB benchmark** demonstrate that TVF-AD achieves state-of-the-art performance while remaining compact, efficient, and robust across different hyperparameter settings.

2 Related Work

2.1 Temporal Time-Series Anomaly Detection Methods

Temporal time-series anomaly detection (TSAD) methods can be broadly categorized into statistical, neural-network-based, and foundation-model-based approaches (Liu & Paparrizos, 2024).

Statistical Methods. Classical approaches detect anomalies through assumptions on local density, distance, normal boundaries, low-dimensional structure, or shape similarity, including LOF (Breunig et al., 2000), KNN (Ramaswamy et al., 2000), MatrixProfile (Yeh et al., 2016), OCSVM (Schölkopf et al., 1999), IForest (Liu et al., 2008), PCA (Aggarwal, 2017), SR (Ren et al., 2019), and KShapeAD (Paparrizos & Gravano, 2015; 2017). While efficient, these methods often rely on handcrafted assumptions and can struggle with complex nonlinear temporal dynamics.

Neural-network-based Methods. Neural approaches learn normal temporal patterns through reconstruction, prediction, or latent representations. Representative methods include AutoEncoder (Sakurada & Yairi, 2014), LSTM-AD (Malhotra et al., 2015), Donut (Xu et al., 2018), OmniAnomaly (Su et al., 2019), and TimesNet (Wu et al., 2023). Transformer-based methods such as TranAD (Tuli et al., 2022) and AnomalyTransformer (Xu et al., 2022) further model long-range dependencies and abnormal associations. Most of these methods, however, remain primarily temporal-domain approaches.

Foundation-model-based Methods. Recent studies exploit large-scale pretrained models for TSAD, including OFA (Zhou et al., 2023), Lag-Llama (Rasul et al., 2023), Chronos (Ansari et al., 2024), TimesFM (Das et al., 2024), and MOMENT (Goswami et al., 2024). These methods benefit from strong pretrained priors and transferability, but typically rely on relatively large backbones and higher computational cost.

2.2 Image-based Time-Series Anomaly Detection Methods

Another line of work transforms time series into image-like representations and applies visual models (Wang & Oates, 2015b; Hatami et al., 2018; Li et al., 2020). Image-based approaches use representations such as GAF (Wang & Oates, 2015a;b); for example, GAF-GAN (Ma & Wang, 2024) performs reconstruction-based detection on transformed images, while ITF-TAD (Namura & Ichikawa, 2024) applies pretrained vision models to wavelet scalograms in a training-free manner. More recent methods such as TAMA (Zhuang et al., 2024) and ViT4TS/VLM4TS (He et al., 2026) further exploit multimodal models. While these studies demonstrate the value of visualized time series, they mainly rely on either image-side detection or relatively heavy multimodal reasoning. In contrast, TVF-AD uses visual representations as a complementary modality within a lightweight temporal–visual fusion framework.

3 Methods

Problem Definition. Given a multivariate time series $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\} \in \mathbb{R}^{T \times D}$, where $\mathbf{x}_t \in \mathbb{R}^D$ is the observation at time step t, multivariate time-series anomaly detection aims to determine whether each time step is normal or anomalous. In the semi-supervised setting, the training set contains only normal observations, while the test set may contain both normal and anomalous observations. The task is to learn an anomaly scoring function $f : \mathbb{R}^{T \times D} \to \mathbb{R}^T$ that outputs a score sequence $\mathbf{s} = \{s_1, s_2, \ldots, s_T\}$, where larger scores indicate higher anomaly likelihood.

Overview. Figure 2 illustrates TVF-AD, a lightweight dual-branch framework for multivariate time-series anomaly detection. Given an input window, the temporal branch encodes the raw sequence, where continuous variables are modeled as temporal features and non-continuous variables modulate them through a FiLM-style mechanism, yielding $\mathbf{z}_i^{ts}$. In parallel, the visual branch converts the same window into CWT-based time–frequency images and extracts aligned visual features offline using a frozen pretrained vision transformer, producing $\mathbf{z}_i^{vis}$. A gating module fuses the two branches into $\mathbf{z}_i^{fused}$. During training, TVF-AD uses a geometry alignment loss to enhance the discriminability of the fused representation for anomaly detection. The pseudocode of training and inference is provided in Appendix A.

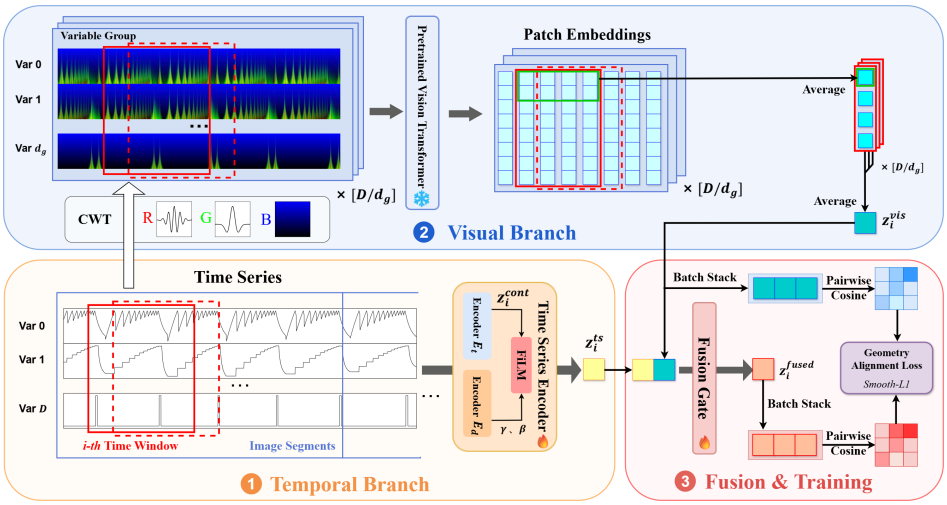


Figure 2: Overall training framework of TVF-AD.

3.1 Temporal Branch

The temporal branch models each multivariate window using a continuous-variable encoding path and a non-continuous state-conditioning path. Given an input window $\mathbf{W}_i \in \mathbb{R}^{L\times D}$, the variables are split into continuous and non-continuous subsets, denoted by $\mathbf{W}_i^{cont} \in \mathbb{R}^{L\times D_{cont}}$ and $\mathbf{W}_i^{disc} \in \mathbb{R}^{L\times D_{disc}}$, respectively. The continuous variables are normalized using training-set statistics and encoded by a lightweight 1D CNN as $\mathbf{z}_i^{\text{cont}} = E_t(\mathbf{W}_i^{\text{cont}})$.

For the non-continuous path, each variable is mapped to discrete IDs, and a transition matrix $\mathbf{T}_i^{(j)} \in \mathbb{R}^{K_j\times K_j}$ is constructed for each non-continuous variable to capture within-window state transitions. The flattened transition matrices are concatenated and encoded by an MLP:

$$\mathbf{z}_i^{\text{disc}} = E_d\Big(\text{concat}\big(\text{vec}(\mathbf{T}_i^{(1)}), \ldots, \text{vec}(\mathbf{T}_i^{(D_{\text{disc}})})\big)\Big).$$

The non-continuous representation is then used to modulate the continuous representation through a FiLM-style transformation:

$$(\boldsymbol{\gamma}_i, \boldsymbol{\beta}_i) = g(\mathbf{z}_i^{\text{disc}}), \quad \mathbf{z}_i^{\text{ts}} = [1 + \alpha \tanh(\boldsymbol{\gamma}_i)] \odot \mathbf{z}_i^{\text{cont}} + \alpha\boldsymbol{\beta}_i.$$

where α is a small scaling factor. The resulting $\mathbf{z}_i^{ts}$ is used for subsequent temporal–visual fusion and representation learning.

3.2 Visual Branch

Following ITF-TAD (Namura & Ichikawa, 2024), we convert each variable into a full-length time–frequency RGB image via continuous wavelet transform (CWT). Specifically, the first two channels are constructed from the log-amplitudes of Morlet and Ricker wavelet responses, while the third channel encodes normalized log-frequency position, providing complementary structural and frequency cues. To incorporate cross-variable information while keeping the visual input compact, we divide the D variables into groups of size d_g, yielding $G = \left\lceil \frac{D}{d_g} \right\rceil$ groups, and concatenate the variable-wise RGB maps within each group along the height dimension to form a composed image. Each full-length composed image is then partitioned along the temporal axis into non-overlapping segments (the blue boxes in Figure 2), resized to the visual encoder input size, and treated as one input image. This design keeps the visual features aligned with the corresponding temporal windows while allowing the visual branch to capture both single-variable time–frequency patterns and local cross-variable structure. Detailed implementation procedures are provided in Appendix B, and the effect of the grouping size d_g is analyzed in Section 4.3.

Suppose the visual encoder tokenizes each segment into a patch grid with H_p rows and W_p columns. Since the variables in a group are stacked vertically, the patch rows corresponding to each variable can be identified accordingly. For each variable, the corresponding H_p/d_g patch rows (highlighted by the green box in Fig. 2) are identified.

To align the visual features with the original time series, let L_{seg} denote the temporal length of each segment. If the visual encoder produces W_p patch columns for one segment, then the temporal size represented by one patch column is $\delta = L_{\text{seg}}/W_p$, which is also used as the stride in the preceding sliding-window construction. Let the anomaly-detection window (highlighted by the red box in Fig. 2) length on the original time series be L. We set $L = q\delta$, where $q \in \mathbb{N}$ is an integer, so that each time window corresponds exactly to q consecutive patch columns.

For the i-th anomaly-detection window, let $\mathcal{C}_i$ denote its aligned set of q patch columns. For each variable j, we first identify its corresponding patch rows $\mathcal{R}_j$. Given the patch tokens $\mathbf{p}_{r,c}$ within the time window region, we first average the tokens of each variable over its corresponding patch rows and aligned patch columns:

$$\mathbf{v}_{i,j} = \frac{1}{|\mathcal{R}_j|\,|\mathcal{C}_i|} \sum_{c\in\mathcal{C}_i} \sum_{r\in\mathcal{R}_j} \mathbf{p}_{r,c},$$

which gives a variable-level visual feature for the j-th variable in the i-th window. We then average the variable-level features over all variables to obtain the final visual representation of the i-th time window:

$$\mathbf{z}_i^{vis} = \frac{1}{D} \sum_{j=1}^{D} \mathbf{v}_{i,j}.$$

The resulting $\mathbf{z}_i^{\text{vis}}$ serves as the window-level visual feature and is directly used by the temporal–visual fusion module.

3.3 Modality Fusion and Training Objective

After obtaining the temporal representation $\mathbf{z}_i^{ts}$ and the offline visual representation $\mathbf{z}_i^{vis}$ for the i-th window, we fuse them through a gating mechanism. Specifically, we first compute a gate vector

$$\mathbf{g}_i = \sigma\big(\mathrm{MLP}_g([\mathbf{z}_i^{ts}; \mathbf{z}_i^{vis}])\big),$$

where $\sigma(\cdot)$ denotes the sigmoid activation and $[\cdot;\cdot]$ denotes feature concatenation. The fused representation is then obtained as

$$\mathbf{z}_i^{fused} = \mathbf{g}_i \odot \mathbf{z}_i^{ts} + (1 - \mathbf{g}_i) \odot \mathbf{z}_i^{vis},$$

where $\odot$ denotes element-wise multiplication. The resulting $\mathbf{z}_i^{fused}$ is used as the final representation for subsequent training and inference.

For training, we adopt a relational geometry alignment objective on the fused representations. Instead of explicitly constructing positive and negative pairs, we encourage the fused representation space to preserve the relational structure of the frozen visual representation.

Given fused features $\{\mathbf{z}_i^{fused}\}$ and their corresponding visual features $\{\mathbf{z}_i^{vis}\}$, we compute pairwise cosine distances as

$$d_{ij}^f = 1 - \cos(\mathbf{z}_i^{fused}, \mathbf{z}_j^{fused}), \qquad d_{ij}^v = 1 - \cos(\mathbf{z}_i^{vis}, \mathbf{z}_j^{vis}).$$

To reduce the effect of different distance scales, both distance matrices are normalized by their mean off-diagonal values. The geometry alignment loss is then defined as

$$\mathcal{L}_{\text{geo}} = \frac{1}{|\mathcal{P}|} \sum_{(i,j)\in\mathcal{P}} \mathrm{SmoothL1}\left(\hat{d}_{ij}^f, \hat{d}_{ij}^v\right),$$

where $\mathcal{P} = \{(i,j) \mid i \neq j\}$ and $\hat{d}_{ij}^f$ and $\hat{d}_{ij}^v$ denote the normalized pairwise distances. By aligning relational geometry rather than individual feature vectors, the fused representation preserves the structural neighborhood information of the frozen visual space while incorporating complementary temporal information. This objective is naturally consistent with the cosine-based memory-bank retrieval used at inference.

3.4 Anomaly Detection via Memory-Bank Retrieval

After training, we build a memory bank $\mathcal{M} = \{\mathbf{z}_j^{tr}\}_{j=1}^{N_m}$ using the fused representations of normal training windows. For each test window $\mathbf{W}_i^{te}$, its representation $\mathbf{z}_i^{te}$ is compared with its Top-k nearest neighbors in $\mathcal{M}$ using cosine similarity, yielding the window-level anomaly score

$$s_i^{\text{win}} = 1 - \frac{1}{k} \sum_{\mathbf{m} \in \mathcal{N}_k(\mathbf{z}_i^{te}, \mathcal{M})} \cos(\mathbf{z}_i^{te}, \mathbf{m}).$$

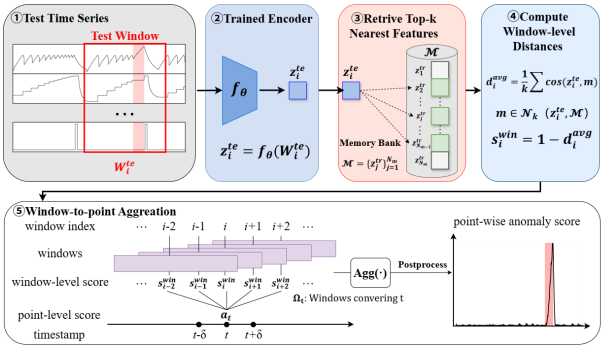


Figure 3: Detection pipeline of TVF-AD.

The window-level scores are then aggregated into a base point-wise anomaly score

$$s_t^{\text{base}} = \text{Agg}\big(\{s_i^{\text{win}} \mid i \in \Omega(t)\}\big),$$

where $\Omega(t)$ denotes the windows covering time step t, and $\text{Agg}(\cdot)$ is implemented using the harmonic mean.

Finally, an offline post-processing procedure further refines s_t^{base} by incorporating multi-scale raw-pattern information, rank-based fusion, and local innovation signals. The rank normalization and innovation calibration are computed over the complete unlabeled test sequence; no anomaly labels are used. Details are provided in Appendix A, Algorithm 3.

4 Experiments

4.1 Experiment Settings

Benchmark. We evaluate TVF-AD on the TSB-AD benchmark (Liu & Paparrizos, 2024), including both the univariate subset TSB-AD-U and the multivariate subset TSB-AD-M under the official evaluation protocol. TSB-AD provides a more reliable and less biased evaluation setting for time-series anomaly detection. More details on the benchmark composition are given in Appendix B.

Implementation Details. TVF-AD adopts a temporal–visual architecture. The temporal branch uses a three-layer 1D CNN for continuous variables and a state-transition encoder with FiLM-style modulation for non-continuous variables. The visual branch converts each window into a CWT-based RGB image and extracts offline features using a frozen CLIP ViT-L/14@336px encoder (Radford et al., 2021). Temporal and visual features are fused by an MLP-based vector gating module. We train the model with AdamW, a learning rate of 1×10^{-4}, and a batch size of 512. Unless otherwise specified, the window size and stride are set to 48 and 8, respectively. Training uses a relational geometry alignment objective based on normalized pairwise cosine distances and SmoothL1 loss, while inference uses top-k memory-bank retrieval ($k = 3$) and harmonic-mean aggregation. Additional implementation details are provided in Appendix B.

Baselines. We compare TVF-AD with statistical, neural, foundation-model-based, and recent representation-based methods.Baseline results are either taken from the original TSB-AD benchmark (Liu & Paparrizos, 2024) and prior work, or reproduced by us under the same TSB-AD evaluation protocol. Unless otherwise specified, reproduced results are averaged over five random seeds.

Evaluation Metrics. Following the official TSB-AD protocol (Liu & Paparrizos, 2024), we report VUS-PR, VUS-ROC (Paparrizos et al., 2022), AUC-PR, AUC-ROC, and Standard-F1. We do not use point adjustment, following recent reliable TSAD evaluation studies. Among these metrics, VUS-PR is used as the primary metric. More details are provided in Appendix B.

4.2 MAIN RESULTS

Table 1: Experimental results on **univariate time-series anomaly detection** in TSB-AD-U. For each measure, the best and second-best values are indicated in **bold** and underlined. All scores are reported with their rankings as "Score/Rank".

Method	VUS-PR	VUS-ROC	AUC-PR	AUC-ROC	Std-F1
(Sub)-PCA† (2017)	0.42/6	0.76/8	0.37/6	0.71/8	0.42/6
KShapeAD† (2017)	0.40/7	0.76/8	0.35/7	0.74/7	0.39/7
DLinear† (2023)	0.25/19	0.74/13	0.21/16	0.62/17	0.26/16
TimesNet§ (2023)	0.26/16	0.72/18	0.18/18	0.61/18	0.24/18
FITS§ (2024)	0.26/16	0.73/17	0.17/19	0.61/18	0.23/19
DADA§ (2025)	0.31/12	0.77/7	0.29/11	0.71/8	0.38/8
KAN-AD§ (2025)	0.43/5	0.82/5	0.41/5	0.80/4	0.44/5
CARLA§ (2025)	0.40/7	0.78/6	0.34/8	0.76/6	0.38/8
PaAno§ (2026)	0.52/2	0.89/2	0.46/2	0.86/2	0.51/2
AnomalyTransformer* (2022)	0.12/22	0.56/22	0.08/22	0.50/22	0.12/22
DCdetector* (2023)	0.09/23	0.56/22	0.05/23	0.50/22	0.10/23
PatchTST* (2023)	0.26/16	0.75/11	0.21/16	0.63/16	0.25/17
iTransformer* (2024)	0.22/21	0.74/13	0.16/20	0.61/18	0.21/21
CrossAD* (2025)	0.43/4	0.83/4	0.42/3	0.77/5	0.44/4
OFA‡ (2023)	0.24/20	0.71/21	0.16/20	0.59/21	0.22/20
Lag-Llama‡ (2023)	0.27/15	0.72/18	0.25/14	0.65/15	0.30/14
MOMENT (FT)‡ (2024)	0.39/9	0.76/8	0.30/9	0.69/10	0.35/10
MOMENT (ZS)‡ (2024)	0.38/10	0.75/11	0.30/9	0.68/11	0.35/10
TimesFM‡ (2024)	0.30/14	0.74/13	0.28/12	0.67/13	0.34/12
ITF-TAD-ResNet‡ (2024)	0.33/11	0.73/16	0.25/13	0.67/12	0.31/13
ITF-TAD-ViT-L‡ (2024)	0.31/13	0.71/20	0.24/15	0.66/14	0.29/15
ViT4TS‡ (2026)	0.49/3	0.88/3	0.41/4	0.85/3	0.45/3
TVF-AD (Ours)	**0.67**/1	**0.95**/1	**0.61**/1	**0.92**/1	**0.64**/1

Note: † statistical methods; § neural-network methods; * Transformer-based neural-network methods; ‡ foundation-model-based methods.

Table 2: Experimental results on **multivariate time-series anomaly detection** in TSB-AD-M. For each measure, the best and second-best values are indicated in **bold** and underlined. All scores are reported with their rankings as "Score/Rank".

Method	VUS-PR	VUS-ROC	AUC-PR	AUC-ROC	Std-F1
PCA† (2017)	0.31/10	0.74/6	0.31/7	0.70/7	0.37/6
DLinear† (2023)	0.29/13	0.70/14	0.27/11	0.66/13	0.32/12
OmniAnomaly§ (2019)	0.27/16	0.73/9	0.23/15	0.68/11	0.32/14
TimesNet§ (2023)	0.19/20	0.64/19	0.13/20	0.56/19	0.20/20
FITS§ (2024)	0.21/18	0.66/18	0.15/18	0.58/18	0.22/18
SARAD§ (2024)	0.36/4	0.74/5	0.31/7	0.70/5	0.38/4
CARLA§ (2025)	0.33/6	0.68/16	0.28/10	0.66/13	0.34/11
CAROTS§ (2025)	0.35/5	0.74/7	0.32/5	0.70/4	0.38/5
DADA§ (2025)	0.31/10	0.73/10	0.31/7	0.69/8	0.35/9
GCAD§ (2025)	0.32/8	0.67/17	0.33/4	0.66/15	0.37/7
KAN-AD§ (2025)	0.41/3	0.75/3	0.38/2	0.73/3	0.42/3
PaAno§ (2026)	0.43/2	0.79/2	0.38/2	0.76/2	0.43/2
AnomalyTransformer* (2022)	0.12/21	0.57/21	0.07/21	0.52/21	0.12/21
DCdetector* (2023)	0.10/22	0.56/22	0.06/22	0.50/22	0.10/22
PatchTST* (2023)	0.28/15	0.71/13	0.26/13	0.65/16	0.32/12
iTransformer* (2024)	0.29/13	0.70/14	0.23/16	0.63/17	0.28/16
CATCH* (2025)	0.30/12	0.73/10	0.24/14	0.67/12	0.30/15
CrossAD* (2025)	0.32/7	0.72/12	0.32/6	0.69/9	0.37/8
OFA‡ (2023)	0.21/18	0.63/20	0.15/18	0.55/20	0.21/19
ITF-TAD-ResNet‡ (2024)	0.32/9	0.75/4	0.26/12	0.70/6	0.35/10
ITF-TAD-ViT-L‡ (2024)	0.22/17	0.74/8	0.16/17	0.68/10	0.22/17
TVF-AD (Ours)	**0.53**/1	**0.84**/1	**0.48**/1	**0.82**/1	**0.52**/1

Note: The same notation as in Table 1 is used.

Tables 1 and 2 report the results on TSB-AD-U and TSB-AD-M, respectively. For clearer comparison, the two recent representation-based methods, **CARLA** and **PaAno**, are highlighted in bold in

the tables. Overall, TVF-AD achieves the best performance on all evaluation metrics across both univariate and multivariate settings, confirming the effectiveness of temporal–visual fusion for time-series anomaly detection.

On TSB-AD-U, TVF-AD ranks first across all five metrics, achieving 0.67 VUS-PR, 0.95 VUS-ROC, 0.61 AUC-PR, 0.92 AUC-ROC, and 0.64 Std-F1. It consistently outperforms the strongest competing baselines across both ranking-based and threshold-based metrics, demonstrating the effectiveness of temporal–visual fusion for univariate anomaly detection.

On TSB-AD-M, TVF-AD again achieves the best results on all five metrics, with 0.53 VUS-PR, 0.84 VUS-ROC, 0.48 AUC-PR, 0.82 AUC-ROC, and 0.52 Std-F1. The consistent gains on both univariate and multivariate benchmarks indicate that the proposed framework learns more discriminative anomaly representations by combining complementary temporal and visual information. Complete experimental results and additional visualization examples are provided in Appendix D.

4.3 Ablation Study

Effect of variable grouping. Table 3 evaluates the number of variables grouped into each image (d_g) on TSB-AD-M. Overall, moderate grouping provides a favorable trade-off between accuracy and efficiency. The setting $d_g = 4$ achieves the best VUS-PR (52.9) and Std-F1 (51.6), and ties for the best AUC-ROC (82.4), while reducing runtime from 360 to 113 minutes compared with $d_g = 1$. Larger group sizes further improve efficiency and can yield higher VUS-ROC, while VUS-PR generally decreases and AUC-PR varies non-monotonically. We therefore adopt $d_g = 4$ as the default setting.

Table 3: Effect of d_g on TSB-AD-M. Metrics are in percentages; runtime (min) excludes metric computation. Best in bold and second-best underlined.

d_g	VUS-PR	VUS-ROC	AUC-PR	AUC-ROC	Std-F1	Time
1	52.3	84.3	**49.4**	82.2	51.1	360
2	52.5	84.1	48.6	82.1	<u>51.5</u>	191
3	<u>52.8</u>	84.1	48.5	82.2	51.3	131
4	**52.9**	84.4	48.4	**82.4**	**51.6**	113
8	52.6	<u>84.8</u>	<u>49.2</u>	**82.4**	51.3	103
12	52.7	**85.0**	49.1	<u>82.3</u>	51.2	<u>89</u>
24	52.1	84.4	47.3	<u>82.3</u>	50.8	**80**

Effect of fusion design. Table 4 compares the single-branch variants with several temporal–visual fusion strategies. On TSB-AD-U, vector gating achieves the best VUS-PR, AUC-PR, and Std-F1, while concatenation with an MLP performs slightly better on VUS-ROC and AUC-ROC. On TSB-AD-M, vector gating again achieves the best VUS-PR, AUC-PR, and Std-F1, whereas scalar gating obtains the highest VUS-ROC and AUC-ROC. Overall, fusion consistently improves over the single-branch variants on the primary anomaly-sensitive metrics, and vector gating provides the strongest and most consistent performance across the two benchmarks.

Table 4: Comparison of alternative fusion mechanisms on TSB-AD-U and TSB-AD-M. Best results are shown in bold and second-best results are underlined.

Dataset	Fusion	Rule	VUS-PR	VUS-ROC	AUC-PR	AUC-ROC	Std-F1
TSB-AD-U	Temporal only	$z = T$	59.03	92.21	53.76	89.28	58.18
	Visual only	$z = V$	49.33	89.75	38.81	86.63	43.19
	Equal average	$z = (T + V)/2$	65.60	94.07	59.31	91.65	<u>63.54</u>
	Concatenation + MLP	$z = \mathrm{MLP}([T; V])$	<u>66.76</u>	**94.95**	<u>59.88</u>	**92.52**	63.17
	Scalar gate	$z = gT + (1 - g)V$	64.42	94.68	57.45	92.24	60.65
	Vector gate	$z = g \odot T + (1 - g) \odot V$	**67.21**	<u>94.85</u>	**60.81**	<u>92.45</u>	**64.25**
TSB-AD-M	Temporal only	$z = T$	45.54	82.87	41.31	80.44	46.57
	Visual only	$z = V$	45.43	86.13	38.72	<u>83.07</u>	44.17
	Equal average	$z = (T + V)/2$	51.44	84.90	<u>48.21</u>	83.03	<u>50.91</u>
	Concatenation + MLP	$z = \mathrm{MLP}([T; V])$	<u>52.32</u>	<u>85.20</u>	46.73	82.89	50.74
	Scalar gate	$z = gT + (1 - g)V$	50.18	**86.37**	43.73	**83.72**	48.79
	Vector gate	$z = g \odot T + (1 - g) \odot V$	**52.93**	84.44	**48.38**	82.43	**51.56**

4.4 Sensitivity Analysis

Fig. 4 presents a representative sensitivity analysis on TSB-AD-M. TVF-AD is generally robust to top-k, batch size, and memory bank ratio, with only limited performance variation. Stride has a rela-

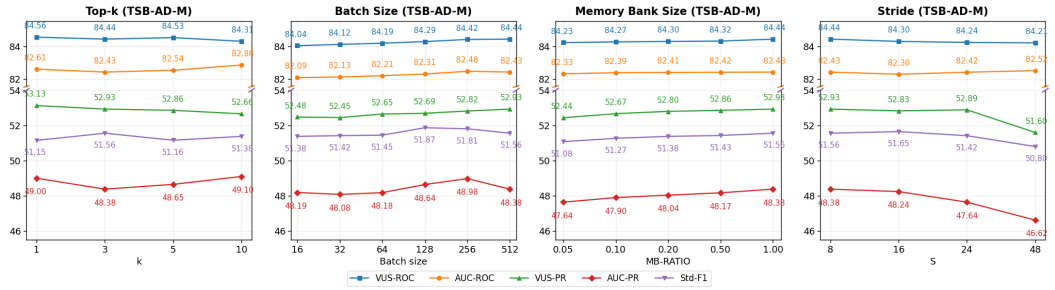


Figure 4: Sensitivity analysis on top-k, batch size, memory bank ratio and stride.

tively larger impact: moderate values yield similar performance, whereas an excessively large stride causes noticeable degradation in VUS-PR, AUC-PR, and Std-F1, likely due to reduced temporal resolution. The mild sensitivity to memory bank compression further suggests that the retrieval module does not require a large memory bank. Overall, TVF-AD requires little hyperparameter tuning, although overly large strides should be avoided. More complete results are provided in Appendix C.

4.5 Efficiency Analysis

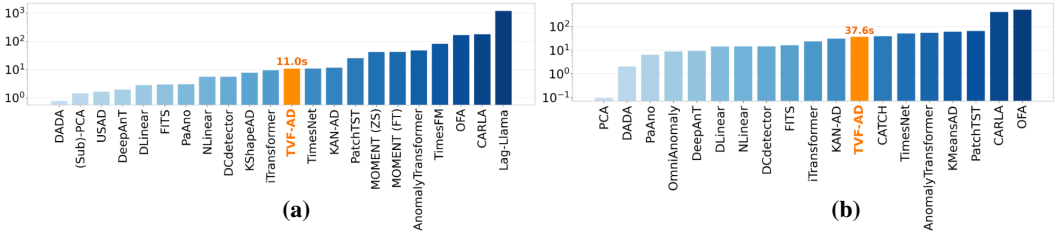


Figure 5: Average runtime on TSB-AD-U and TSB-AD-M excluding metric computation. (a) Results on TSB-AD-U. (b) Results on TSB-AD-M.

Figure 5 compares the average runtime of TVF-AD with representative baselines on TSB-AD-U and TSB-AD-M. Overall, TVF-AD demonstrates favorable efficiency, with runtime comparable to several mainstream methods while benefiting from temporal–visual fusion. The model is also relatively compact, with only about 6.9M trainable parameters. In addition, runtime can be further reduced by moderately increasing the variable grouping size d_g and the stride, which decrease the numbers of composed images.

5 Conclusion

We proposed TVF-AD, a temporal–visual framework for time-series anomaly detection. By introducing a complementary visual view, TVF-AD improves anomaly representation learning while preserving efficiency through offline visual feature extraction and a lightweight temporal branch. Extensive experiments on the TSB-AD benchmark show that TVF-AD achieves state-of-the-art performance. Ablation and sensitivity studies further confirm the effectiveness of key designs, including temporal–visual fusion and variable grouping in the visual branch. One limitation of the current framework is that it is primarily designed for offline detection, since the visual representations are pre-computed from time–frequency images and the memory bank remains fixed during inference. As a result, the current model does not explicitly adapt to evolving normal patterns or temporal distribution shifts that are unseen during training. Extending TVF-AD to streaming scenarios through incremental visual encoding, adaptive representation learning, and dynamic memory-bank updates is an interesting direction for future work.

AI USE STATEMENT

We used generative AI tools to assist with language editing, formatting checks, literature search, drafting portions of the manuscript, and code implementation. The authors reviewed and revised AI-assisted drafts, checked suggested references against their original sources, and manually reviewed the AI-assisted code for correctness and consistency with the intended methodology. All reported experimental results were obtained by running the implementation, rather than generated by AI tools. Generative AI tools were not used to generate synthetic datasets or prove mathematical claims. The authors take full responsibility for the final text, references, code, experimental results, and conclusions.

REPRODUCIBILITY STATEMENT

We provide details of TVF-AD to support reproducibility. Appendix A presents pseudocode for offline visual feature extraction, model training, and inference. Appendix B describes the benchmark datasets, evaluation metrics, implementation settings, and time-series-to-image construction procedure. Appendix C reports sensitivity analyses and ablation studies of the main design choices. Appendix D provides subset-wise results with means and standard deviations over five random seeds, as well as details of the computational environment and runtime evaluation. The inference description also specifies the use of the complete unlabeled test sequence for offline post-processing.

ETHICS STATEMENT

This work studies time-series anomaly detection using existing benchmark datasets. Although anomaly detection can support monitoring applications, false positives may trigger unnecessary interventions, while false negatives may leave important events undetected. Performance on the evaluated benchmarks does not establish reliability in safety-critical deployments. Practical use should include application-specific validation, human oversight, and appropriate safeguards for any sensitive monitoring data. The offline evaluation setting should also be taken into account when considering deployment in real-time systems.

REFERENCES

Charu C. Aggarwal. *Outlier Analysis*. Springer, Cham, Switzerland, 2nd edition, 2017. doi: 10.1007/978-3-319-47578-3.

Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262:134–147, 2017.

Chuadhry Mujeeb Ahmed, Venkata Reddy Palleti, and Aditya P Mathur. Wadi: A water distribution testbed for research in the design of secure cyber physical systems. In *Proceedings of the 3rd international workshop on cyber-physical systems for smart water networks*, pp. 25–28, 2017.

Abdul Fatir Ansari, Lorenzo Stella, Ali Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Bernie Wang. Chronos: Learning the Language of Time Series. *Transactions on Machine Learning Research*, 2024. URL https://mlanthology.org/tmlr/2024/ansari2024tmlr-chronos/.

Marc Bächlin, Meir Plotnik, Daniel Roggen, Inbal Maidan, Jeffrey M Hausdorff, Nir Giladi, and Gerhard Tröster. Wearable assistant for parkinson's disease patients with the freezing of gait symptom. *IEEE Transactions on Information Technology in Biomedicine*, 14(2):436–446, 2010. doi: 10.1109/TITB.2009.2036165.

P. Boniol, J. Paparrizos, T. Palpanas, and M. J. Franklin. Sand: Streaming subsequence anomaly detection. *PVLDB*, 14(10), 2021.

Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. LOF: identifying density-based local outliers. In Weidong Chen, Jeffrey F. Naughton, and Philip A. Bernstein (eds.), *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, pp. 93–104. ACM, 2000. doi: 10.1145/342009.335388. URL https://doi.org/10.1145/342009.335388.

Mouxiang Chen, Lefei Shen, Zhuo Li, Xiaoyun Joy Wang, Jianling Sun, and Chenghao Liu. VisionTS: Visual masked autoencoders are free-lunch zero-shot time series forecasters. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 8979–9007. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/chen25be.html.

Richard Crout, Dawn Petraitis, and NOAA National Data Buoy Center. Physical and meteorological data from the tropical atmosphere ocean (TAO) array in the tropical pacific ocean, 2012. URL https://www.ncei.noaa.gov/archive/accession/NDBC-TAO-Buoy. Dataset.

Zhihao Dai, Ligang He, Shuanghua Yang, and Matthew Leeke. SARAD: Spatial association-aware anomaly detection and diagnosis for multivariate time series. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=gmf5Aj01Hz.

Zahra Zamanzadeh Darban, Geoffrey I Webb, Shirui Pan, Charu C Aggarwal, and Mahsa Salehi. Carla: Self-supervised contrastive representation learning for time series anomaly detection. *Pattern Recognition*, 157:110874, 2025.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 10148–10167. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/das24c.html.

Ailin Deng and Bryan Hooi. Graph neural network-based anomaly detection in multivariate time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 4027–4035, 2021.

J-P Eckmann, S Oliffson Kamphorst, and David Ruelle. Recurrence plots of dynamical systems. In *Turbulence, strange attractors and chaos*, pp. 441–445. World Scientific, 1995.

Pavel Filonov, Andrey Lavrentyev, and Artem Vorontsov. Multivariate industrial time series with cyber-attack simulation: Fault detection using an lstm-based predictive data model. *arXiv preprint arXiv:1612.06676*, 2016.

Patrick Fleith. Controlled anomalies time series (cats) dataset, 2023.

Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220, 2000.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. Moment: A family of open time-series foundation models. In *International Conference on Machine Learning*, 2024.

Scott David Greenwald, Ramesh S Patil, and Roger G Mark. *Improved detection and classification of arrhythmias in noise-corrupted electrocardiograms using contextual information*. IEEE, 1990.

Nima Hatami, Yann Gavet, and Johan Debayle. Classification of time-series images using deep convolutional neural networks. In Antanas Verikas, Petia Radeva, Dmitry Nikolaev, and Jianhong Zhou (eds.), *Tenth International Conference on Machine Vision (ICMV 2017)*, volume 10696, pp. 106960Y. International Society for Optics and Photonics, SPIE, 2018. doi: 10.1117/12.2309486. URL https://doi.org/10.1117/12.2309486.

Zelin He, Sarah Alnegheimish, and Matthew Reimherr. Harnessing vision-language models for time series anomaly detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 21690–21698, 2026. doi: 10.1609/aaai.v40i26.39319.

Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 387–395, 2018.

IOPS. IOPS dataset. http://iops.ai/dataset_detail/?id=10. Accessed: 2026-05-04.

Vincent Jacob, Fei Song, Arnaud Stiegler, Bijan Rad, Yanlei Diao, and Nesime Tatbul. Exathlon: A benchmark for explainable anomaly detection over time series. *Proceedings of the VLDB Endowment*, 14(11):2613–2626, 2021. doi: 10.14778/3476249.3476307.

Hyungi Kim, Jisoo Mok, Dongjun Lee, Jaihyun Lew, Sungjae Kim, and Sungroh Yoon. Causality-aware contrastive learning for robust multivariate time-series anomaly detection. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 30591–30608. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/kim25aa.html.

Kwei-Herng Lai, Daochen Zha, Junjie Xu, Yue Zhao, Guanchu Wang, and Xia Hu. Revisiting time series outlier detection: Definitions and benchmarks. In J. Vanschoren and S. Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/ec5decca5ed3d6b8079e2e7e7bacc9f2-Paper-round1.pdf.

N Laptev, S Amizadeh, and Y Billawala. S5-a labeled anomaly detection dataset, version 1.0 (16m). *URL https://webscope. sandbox. yahoo. com/catalog. php*, 2015.

Beibu Li, Qichao Shentu, Yang Shu, Hui Zhang, Ming Li, Ning Jin, Bin Yang, and Chenjuan Guo. Crossad: Time series anomaly detection with cross-scale associations and cross-window modeling. *Advances in Neural Information Processing Systems*, 2025.

Xixi Li, Yanfei Kang, and Feng Li. Forecasting with time series imaging. *Expert Systems with Applications*, 160:113680, 2020. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2020.113680. URL https://www.sciencedirect.com/science/article/pii/S0957417420305042.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422, 2008. doi: 10.1109/ICDM.2008.17.

Qinghua Liu and John Paparrizos. The elephant in the room: Towards a reliable time-series anomaly detection benchmark. In *NeurIPS 2024*, 2024.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 11116–11140, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/2ea18fdc667e0ef2ad82b2b4d65147ad-Paper-Conference.pdf.

Zehao Liu, Mengzhou Gao, and Pengfei Jiao. Gcad: Anomaly detection in multivariate time series from the perspective of granger causality. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 19041–19049, 2025.

Jidong Ma and Hairu Wang. Anomaly detection in sensor data via encoding time series into images. *Journal of King Saud University - Computer and Information Sciences*, 36(10):102232, 2024. ISSN 1319-1578. doi: https://doi.org/10.1016/j.jksuci.2024.102232. URL https://www.sciencedirect.com/science/article/pii/S1319157824003215.

Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. Long short term memory networks for anomaly detection in time series. In *Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pp. 89–94, 2015.

Aditya P Mathur and Nils Ole Tippenhauer. Swat: A water treatment testbed for research and training on ics security. In *2016 international workshop on cyber-physical systems for smart water networks (CySWater)*, pp. 31–36. IEEE, 2016.

Steffen Moritz, Frederik Rehbach, Sowmya Chandrasekaran, Margarita Rebolledo, and Thomas Bartz-Beielstein. Gecco industrial challenge 2018 dataset: A water quality dataset for the internet of things: Online anomaly detection for drinking water quality competition at the genetic and evolutionary computation conference 2018, kyoto, japan, 2018. Dataset.

Nobuo Namura and Yuma Ichikawa. Training-free time-series anomaly detection: Leveraging image foundation models. *arXiv preprint arXiv:2408.14756*, 2024.

Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*, 2023.

John Paparrizos and Luis Gravano. k-Shape: Efficient and Accurate Clustering of Time Series. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pp. 1855–1870, 2015.

John Paparrizos and Luis Gravano. Fast and Accurate Time-Series Clustering. *ACM Transactions on Database Systems (ACM TODS)*, 42(2):1–49, 2017.

John Paparrizos, Paul Boniol, Themis Palpanas, Ruey S Tsay, Aaron Elmore, and Michael J Franklin. Volume Under the Surface: A New Accuracy Evaluation Measure for Time-Series Anomaly Detection. *Proceedings of the VLDB Endowment*, 15(11):2774–2787, 2022.

Jinju Park and Seokho Kang. Paano: Patch-based representation learning for time-series anomaly detection. In *Proceedings of International Conference on Learning Representations*, 2026.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 427–438, 2000.

Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Arian Khorasani, George Adamopoulos, Rishika Bhagwatkar, Marin Biloš, Hena Ghonia, Nadhir Hassen, Anderson Schneider, Sahil Garg, Alexandre Drouin, Nicolas Chapados, Yuriy Nevmyvaka, and Irina Rish. Lag-Llama: Towards Foundation Models for Time Series Forecasting. In *NeurIPS 2023 Workshops: R0-FoMo*, 2023. URL https://mlanthology.org/neuripsw/2023/rasul2023neuripsw-lagllama/.

Hansheng Ren, Bixiong Xu, Yujing Wang, Chao Yi, Congrui Huang, Xiaoyu Kou, Tony Xing, Mao Yang, Jie Tong, and Qi Zhang. Time-series anomaly detection service at microsoft. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pp. 3009–3017, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330680. URL https://doi.org/10.1145/3292500.3330680.

Daniel Roggen, Alberto Calatroni, Mirco Rossi, Thomas Holleczek, Kilian Förster, Gerhard Tröster, Paul Lukowicz, David Bannach, Gerald Pirkl, Alois Ferscha, et al. Collecting complex activity datasets in highly rich networked sensor environments. In *2010 Seventh international conference on networked sensing systems (INSS)*, pp. 233–240. IEEE, 2010.

Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, MLSDA'14, pp. 4–11, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450331593. doi: 10.1145/2689746.2689747. URL https://doi.org/10.1145/2689746.2689747.

Bernhard Schölkopf, Robert C Williamson, Alex Smola, John Shawe-Taylor, and John Platt. Support vector method for novelty detection. In S. Solla, T. Leen, and K. Müller (eds.), *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999. URL https://proceedings.neurips.cc/paper_files/paper/1999/file/8725fb777f25776ffa9076e44fcfd776-Paper.pdf.

Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy - ICISSP*, pp. 108–116. INSTICC, SciTePress, 2018. ISBN 978-989-758-282-0. doi: 10.5220/0006639801080116.

Qichao Shentu, Beibu Li, Kai Zhao, Yang Shu, Zhongwen Rao, Lujia Pan, Bin Yang, and Chenjuan Guo. Towards a general time series anomaly detector with adaptive bottlenecks and dual adversarial decoders. In *International Conference on Learning Representations*, volume 2025, pp. 81358–81381, 2025.

Haotian Si, Jianhui Li, Changhua Pei, Hang Cui, Jingwen Yang, Yongqian Sun, Shenglin Zhang, Jingjing Li, Haiming Zhang, Jing Han, et al. Timeseriesbench: An industrial-grade benchmark for time series anomaly detection models. In *2024 IEEE 35th International Symposium on Software Reliability Engineering (ISSRE)*, pp. 61–72. IEEE, 2024.

Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2828–2837, 2019.

Markus Thill, Wolfgang Konen, and Thomas Bäck. The mackey-glass anomaly benchmark (MGAB), 2020.

Luan Tran, Liyue Fan, and Cyrus Shahabi. Distance-based outlier detection in data streams. *Proceedings of the VLDB Endowment*, 9(12):1089–1100, 2016.

Shreshth Tuli, Giuliano Casale, and Nicholas R Jennings. TranAD: Deep Transformer Networks for Anomaly Detection in Multivariate Time Series Data. *Proceedings of VLDB*, 15(6):1201–1214, 2022.

Alexander von Birgelen and Oliver Niggemann. *Anomaly Detection and Localization for Cyber-Physical Production Systems with Self-Organizing Maps*, pp. 55–71. Springer Berlin Heidelberg, Berlin, Heidelberg, 2018. doi: 10.1007/978-3-662-57805-6_4. URL https://doi.org/10.1007/978-3-662-57805-6_4.

Zhiguang Wang and Tim Oates. Encoding time series as images for visual inspection and classification using tiled convolutional neural networks. In *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015a. URL http://coral-lab.umbc.edu/wp-content/uploads/2015/05/10179-43348-1-SM1.pdf.

Zhiguang Wang and Tim Oates. Imaging time-series to improve classification and imputation. In Qiang Yang and Michael J. Wooldridge (eds.), *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pp. 3939–3945. AAAI Press, 2015b. URL http://ijcai.org/Abstract/15/553.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *International Conference on Learning Representations*, 2023.

Renjie Wu and Eamonn Keogh. Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress. *IEEE Transactions on Knowledge and Data Engineering*, 35 (3):2421–2429, 2023. doi: 10.1109/TKDE.2021.3112126.

Xingjian Wu, Xiangfei Qiu, Zhengyu Li, Yihang Wang, Jilin Hu, Chenjuan Guo, Hui Xiong, and Bin Yang. CATCH: Channel-aware multivariate time series anomaly detection via frequency patching. In *International Conference on Learning Representations*, 2025.

Haowen Xu, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, Ying Liu, Youjian Zhao, Dan Pei, Yang Feng, et al. Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pp. 187–196. International World Wide Web Conferences Steering Committee, 2018.

Jiehui Xu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Anomaly transformer: Time series anomaly detection with association discrepancy. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=LzQQ89U1qm_.

Zhijian Xu, Ailing Zeng, and Qiang Xu. Fits: Modeling time series with 10k parameters. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 26295–26318, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/701251e1db4a2e4dd2ef23f5265d5936-Paper-Conference.pdf.

Yiyuan Yang, Chaoli Zhang, Tian Zhou, Qingsong Wen, and Liang Sun. DCdetector: Dual attention contrastive representation learning for time series anomaly detection. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3033–3045, 2023. doi: 10.1145/3580305.3599295.

Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. Matrix profile i: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pp. 1317–1322, 2016. doi: 10.1109/ICDM.2016.0179.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

Chuxu Zhang, Dongjin Song, Yuncong Chen, Xinyang Feng, Cristian Lumezanu, Wei Cheng, Jingchao Ni, Bo Zong, Haifeng Chen, and Nitesh V Chawla. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 1409–1416, 2019.

Jiuqi Elise Zhang, Di Wu, and Benoit Boulet. Time series anomaly detection for smart grids: A survey. In *2021 IEEE Electrical Power and Energy Conference (EPEC)*, pp. 125–130, 2021. doi: 10.1109/EPEC52095.2021.9621752.

Shenglin Zhang, Zhenyu Zhong, Dongwen Li, Qiliang Fan, Yongqian Sun, Man Zhu, Yuzhi Zhang, Dan Pei, Jiyan Sun, Yinlong Liu, et al. Efficient kpi anomaly detection through transfer learning for large-scale web services. *IEEE Journal on Selected Areas in Communications*, 40(8):2440–2455, 2022.

Quan Zhou, Changhua Pei, Fei Sun, Han Jing, Zhengwei Gao, Haiming Zhang, Gaogang Xie, Dan Pei, and Jianhui Li. KAN-AD: Time series anomaly detection with kolmogorov–arnold networks. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 79136–79149. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/zhou25u.html.

Tian Zhou, Peisong Niu, Xue Wang, Liang Sun, and Rong Jin. One Fits All: Power general time series analysis by pretrained LM. In *Advances in Neural Information Processing Systems*, volume 36, pp. 43322–43355, 2023.

Jiaxin Zhuang, Leon Yan, Zhenwei Zhang, Ruiqi Wang, Jiawei Zhang, and Yuantao Gu. See it, think it, sorted: Large multimodal models are few-shot time series anomaly analyzers. *arXiv preprint arXiv:2411.02465*, 2024.

A PSEUDOCODE

Algorithm 1 Offline Visual Feature Extraction

Require: Multivariate time series $\mathbf{X} \in \mathbb{R}^{T \times D}$, CWT parameter n, segment length L_{seg}, window length W, window stride s, group size d_g, frozen vision encoder E_v

Ensure: Window-aligned visual representations $\mathbf{Z}^{vis} \in \mathbb{R}^{M' \times C}$

Generate primarily non-overlapping temporal spans $\{(l_m, r_m)\}_{m=1}^{M}$ of length L_{seg}
Add a tail-aligned span when necessary to cover the suffix
Pre-compute valid patch-column indices for each span to remove duplicated temporal coverage
Initialize per-variable feature buffers $\mathcal{B}_1, \ldots, \mathcal{B}_D \leftarrow \emptyset$
for each variable group $g = \{i_1, \ldots, i_{d_g}\}$ **do**
    for each variable $i \in g$ **do**
        Convert the full sequence $\mathbf{X}_{:,i}$ into a full RGB time–frequency map by CWT
        Normalize the R/G channels using statistics estimated only from the training split
    end for
    for each span (l_m, r_m) **do**
        Crop the RGB segment of each variable in g over $[l_m, r_m)$
        Concatenate the variable-wise RGB segments along the height dimension
        Resize the composed image and feed it to the frozen encoder E_v
        Obtain a patch-token grid with W_p patch columns
        Average the patch rows spatially associated with each variable
        Keep only valid patch columns and append them to the corresponding buffers $\mathcal{B}_i$
    end for
end for
Stack all variable buffers into $\mathbf{F}_{\text{col}} \in \mathbb{R}^{N \times D \times C}$
Compute the temporal coverage of one patch column:

$$\delta \leftarrow \frac{L_{\text{seg}}}{W_p}$$

Verify that L_{seg}, W, and s are divisible by W_p, δ, and δ, respectively
Convert the window parameters into patch-column units:

$$q \leftarrow \frac{W}{\delta}, \qquad s_{\text{col}} \leftarrow \frac{s}{\delta}$$

Aggregate every q consecutive feature columns in $\mathbf{F}_{\text{col}}$ by mean pooling with stride s_{col}, obtaining variable-level window features $\mathbf{F} \in \mathbb{R}^{M' \times D \times C}$
for $i = 1$ to M' **do**
    Average the variable-level features to obtain the window-level visual representation:

$$\mathbf{z}_i^{vis} \leftarrow \frac{1}{D} \sum_{j=1}^{D} \mathbf{F}_{i,j,:}$$

end for
Stack all window-level visual representations into $\mathbf{Z}^{vis} \leftarrow [\mathbf{z}_1^{vis}, \ldots, \mathbf{z}_{M'}^{vis}]^\top \in \mathbb{R}^{M' \times C}$
return $\mathbf{Z}^{vis}$

Algorithm 2 Training Stage of TVF-AD

Require: Training dataset $\mathcal{D}_{\text{tr}}$, number of iterations N_{iter}, learning rate η
Ensure: Best checkpoint θ^*

Build temporal encoder E_t, optional discrete encoder E_d, FiLM modulator M_f, and fusion module G_f
Initialize optimizer and gradient scaler
$\mathcal{L}_{\text{best}} \leftarrow +\infty$
for $t = 1$ to N_{iter} **do**
    Sample a mini-batch $\{(\mathbf{x}_i, \mathbf{z}_i^{vis}, \mathbf{T}_i)\}_{i=1}^{B}$
    Compute fused representations $\{\mathbf{z}_i^{fused}\}_{i=1}^{B}$ using EXTRACTFUSIONEMBED
    Compute pairwise cosine-distance matrices in the fused and frozen visual spaces:

$$D_{ij}^{f} = 1 - \cos(\mathbf{z}_i^{fused}, \mathbf{z}_j^{fused}), \qquad D_{ij}^{v} = 1 - \cos(\mathbf{z}_i^{vis}, \mathbf{z}_j^{vis})$$

    Normalize $\mathbf{D}^f$ and $\mathbf{D}^v$ by their mean off-diagonal distances
    Compute the geometry alignment loss

$$\mathcal{L}_{\text{geo}} = \frac{1}{|\mathcal{P}|} \sum_{(i,j)\in\mathcal{P}} \text{SmoothL1}\left(\widehat{D}_{ij}^{f}, \widehat{D}_{ij}^{v}\right), \quad \mathcal{P} = \{(i,j) \mid i \neq j\}$$

    Backpropagate $\mathcal{L}_{\text{geo}}$ and update model parameters
    if $\mathcal{L}_{\text{geo}} < \mathcal{L}_{\text{best}}$ **then**
        $\mathcal{L}_{\text{best}} \leftarrow \mathcal{L}_{\text{geo}}$
        Save current model as θ^*
    end if
end for
return θ^*

function EXTRACTFUSIONEMBED($\mathbf{x}, \mathbf{z}^{vis}, \mathbf{T}$)
    $\mathbf{z}^{cont} \leftarrow E_t(\mathbf{x})$
    if E_d, M_f, and $\mathbf{T}$ are available **then**
        $\mathbf{z}^{disc} \leftarrow E_d(\mathbf{T})$
        $\mathbf{z}^{ts} \leftarrow M_f(\mathbf{z}^{cont}, \mathbf{z}^{disc})$
    else
        $\mathbf{z}^{ts} \leftarrow \mathbf{z}^{cont}$
    end if
    return $G_f(\mathbf{z}^{ts}, \mathbf{z}^{vis})$
end function

Algorithm 3 Inference with Memory-Bank Retrieval and Post-processing

Require: Best checkpoint θ^*, training-bank dataset $\mathcal{D}_{\text{bank}}$, test dataset $\mathcal{D}_{\text{te}}$, raw time series $\mathbf{X}$, training length T_{tr}, Top-k, base window size W, post-processing scale factor ρ, aggregation rule Agg

Ensure: Final point-wise anomaly scores $\mathbf{s}^{\text{final}}$

Load θ^* and switch all modules to evaluation mode

Step 1: Build normal feature memory bank

Initialize $\mathcal{M} \leftarrow \emptyset$

for each batch in $\mathcal{D}_{\text{bank}}$ **do**

  Compute fused window embeddings $\mathbf{z}$ using EXTRACTFUSIONEMBED

  Append $\mathbf{z}$ to $\mathcal{M}$

end for

Step 2: Extract test-window embeddings

Initialize $\mathcal{Z}_{\text{te}} \leftarrow \emptyset$

for each batch in $\mathcal{D}_{\text{te}}$ **do**

  Compute test embeddings $\mathbf{z}^{te}$ using EXTRACTFUSIONEMBED

  Append $\mathbf{z}^{te}$ to $\mathcal{Z}_{\text{te}}$

end for

Step 3: Memory-bank retrieval and window scoring

for each test embedding $\mathbf{z}_i^{te} \in \mathcal{Z}_{\text{te}}$ **do**

  Retrieve its Top-k cosine neighbors from $\mathcal{M}$

  Compute $s_i^{\text{win}} \leftarrow 1 - \dfrac{1}{k} \displaystyle\sum_{\mathbf{m} \in \mathcal{N}_k(\mathbf{z}_i^{te}, \mathcal{M})} \cos(\mathbf{z}_i^{te}, \mathbf{m})$

end for

Step 4: Window-to-point aggregation

for each time step t **do**

  $s_t^{\text{base}} \leftarrow \mathrm{Agg}\left(\{s_i^{\text{win}} \mid i \in \Omega(t)\}\right)$

end for

Step 5: Post-processing with three auxiliary signals

Normalize the raw time series using the mean and standard deviation of the training prefix

Compute short- and long-scale raw-pattern anomaly scores using window lengths $w_s = W/\rho$ and $w_l = \rho W$, where each query window is matched to its nearest normal pattern in the training prefix

Obtain the multi-scale raw-pattern signal by averaging the two scores: $s_t^{\text{nei}} \leftarrow \dfrac{1}{2}\left(s_t^{\text{raw},w_s} + s_t^{\text{raw},w_l}\right)$

Compute the local innovation signal using a median filter of width 5: $I_t \leftarrow |\widetilde{x}_t - \mathrm{MedFilt}_5(\widetilde{\mathbf{x}})_t|$

Rank-normalize the base score and the multi-scale raw-pattern score, and fuse them with power sharpening: $s_t^{\text{fuse}} \leftarrow \dfrac{1}{2}\left(\dfrac{\mathrm{rank}(s_t^{\text{base}})}{T}\right)^p + \dfrac{1}{2}\left(\dfrac{\mathrm{rank}(s_t^{\text{nei}})}{T}\right)^p, \quad p = 12$

Compute the innovation threshold $\tau \leftarrow \max\left(\alpha \max\limits_{t \le T_{\text{tr}}} I_t,\, Q_{1-\beta}\left(\{I_t : t > T_{\text{tr}}\}\right)\right)$, where $\alpha = 1.4$ and $\beta = 0.005$

Set $s_t^{\text{final}} \leftarrow 1$ if $I_t > \tau$; otherwise set $s_t^{\text{final}} \leftarrow s_t^{\text{fuse}}$

return $\mathbf{s}^{\text{final}}$

B Evaluation Details

B.1 Benchmark Details

Table 5: Subset composition of the TSB-AD benchmark.

Category	Subsets (# Time-Series)
TSB-AD-U (Eval)	UCR (70), YAHOO (30), WSD (20), CATSv2 (1), Daphnet (1), Exathlon (30), IOPS (15), LTDB (8), MGAB (8), MITDB (7), MSL (7), NAB (23), NEK (8), OPPORTUNITY (27), Power (1), SED (2), SMAP (17), SMD (33), Stock (8), SVDB (18), SWaT (1), TAO (2), TODS (13)
TSB-AD-M (Eval)	CATSv2 (5), CreditCard (1), Daphnet (1), Exathlon (25), GECCO (1), Genesis (1), GHL (23), LTDB (4), MITDB (11), MSL (14), OPPORTUNITY (7), PSM (1), SMAP (25), SMD (20), SVDB (28), SWaT (2), TAO (11)

TSB-AD collected 13 univariate datasets and 20 multivariate datasets in the initial stage. After dataset curation, the benchmark finally comprised 23 univariate datasets, including 12 derived from multivariate datasets, and 17 multivariate datasets. Table 5 summarizes the subset composition of TSB Benchmark. TSB-AD includes the following datasets:

- **CATSv2 Fleith (2023)** is the second release of the Controlled Anomalies Time Series dataset, containing commands, external stimuli, and telemetry signals from a simulated complex dynamical system with 200 injected anomalies.
- **CreditCard Sharafaldin et al. (2018)** is an intrusion detection evaluation dataset composed of labeled network flows, together with full packet payloads in pcap format, corresponding profiles, and annotation information.
- **Daphnet Bächlin et al. (2010)** contains annotated signals from three acceleration sensors placed on the hip and leg of Parkinson's disease patients, and is designed to capture freezing of gait (FoG) events during walking.
- **Exathlon Jacob et al. (2021)** is built from real traces collected from a Spark cluster over a period of 2.5 months. For each anomaly instance, labels are provided for both the root-cause interval and the corresponding effect interval.
- **GECCO Moritz et al. (2018)** is a water quality dataset introduced for a competition on online anomaly detection in drinking water systems.
- **Genesis von Birgelen & Niggemann (2018)** is a portable pick-and-place demonstrator that relies on an air tank to supply all gripping and storage units.
- **GHL Filonov et al. (2016)** records the states of three reservoirs, such as temperature and level, where anomalies correspond to changes in maximum temperature or pump frequency.
- **IOPS IOPS** is a dataset of performance indicators that describe the scale, service quality, and machine health status of web services.
- **LTDB Goldberger et al. (2000)** is a collection of seven long-duration ECG recordings, each lasting from 14 to 22 hours, with manually reviewed beat annotations.
- **MGAB Thill et al. (2020)** is based on the Mackey–Glass time series, in which anomalous patterns exhibit chaotic behaviors that are difficult for humans to distinguish visually.
- **MITDB Goldberger et al. (2000)** contains 48 half-hour excerpts of two-channel ambulatory ECG recordings collected from 47 subjects by the BIH Arrhythmia Laboratory between 1975 and 1979.
- **MSL Hundman et al. (2018)** is a dataset collected from the Curiosity Rover mission on Mars.
- **NAB Ahmad et al. (2017)** consists of labeled real-world and artificial time series, including AWS server metrics, online advertisement click rates, real-time traffic data, and Twitter mention streams of major publicly traded companies.
- **NEK Si et al. (2024)** is collected from real production network equipment.

- **OPPORTUNITY Roggen et al. (2010)** is a benchmark for human activity recognition, covering tasks such as classification, automatic segmentation, sensor fusion, and feature extraction, based on motion sensor readings collected during daily activities.
- **SED Boniol et al. (2021)** is a simulated engine disk dataset from the NASA Rotatory Dynamics Laboratory, representing disk revolutions recorded over multiple runs at a speed of 3000 rpm.
- **SMAP Hundman et al. (2018)** contains real spacecraft telemetry data from the Soil Moisture Active Passive satellite. In each time series, one variable corresponds to a sensor measurement, while the remaining variables are binary encoded commands.
- **SMD Su et al. (2019)** is a five-week dataset collected from a large Internet company, including three groups of entities from 28 different machines.
- **Stock Tran et al. (2016)** is a stock trading trace dataset containing one million transaction records collected during the trading hours of a day.
- **SVDB Greenwald et al. (1990)** contains 78 half-hour ECG recordings selected to complement the supraventricular arrhythmia examples in the MIT-BIH Arrhythmia Database.
- **SWaT Mathur & Tippenhauer (2016)** is a secure water treatment dataset collected from 51 sensors and actuators, where anomalies correspond to abnormal behaviors under attack scenarios.
- **TAO Crout et al. (2012)** contains 575,648 records with three attributes collected from the Tropical Atmosphere Ocean project.
- **TODS Lai et al. (2021)** is a synthetic dataset designed to include multiple anomaly types, such as global, contextual, shapelet, seasonal, and trend anomalies.
- **UCR Wu & Keogh (2023)** is a collection of univariate time series drawn from multiple domains, including air temperature, arterial blood pressure, astronomy, ECG, gait, power demand, respiration, and walking accelerometer data. Most anomalies are artificially introduced.
- **WADI Ahmed et al. (2017)** is a water distribution dataset collected from 123 sensors and actuators over 16 days of continuous operation.
- **WSD Zhang et al. (2022)** is a web service dataset containing real-world KPIs collected from large Internet companies.
- **YAHOO Laptev et al. (2015)** is a dataset released by Yahoo Labs, consisting of both real and synthetic time series derived from the production traffic of several Yahoo services.

B.2 Performance Measures

Notations. Consider a test time series $\mathbf{X}' = (\mathbf{x}_1, \ldots, \mathbf{x}_{N'})$ with N' time steps, where each observation $\mathbf{x}_t \in \mathbb{R}^d$ is a d-dimensional vector at time step t. Let $y_t \in \{0, 1\}$ be the corresponding ground-truth label, where $y_t = 1$ denotes an anomalous point and $y_t = 0$ otherwise. Let $s_t \in \mathbb{R}$ denote the anomaly score predicted for time step t. Given a threshold τ, the corresponding binary prediction is defined as $\hat{y}_t(\tau) = \mathbf{1}(s_t \geq \tau)$.

AUC-ROC. The ROC curve characterizes the trade-off between the true positive rate and the false positive rate under different thresholds τ. Specifically, for a given threshold τ, the true positive rate (TPR) and false positive rate (FPR) are given by

$$\mathrm{TPR}(\tau) = \frac{\sum_{t=1}^{N'} \hat{y}_t(\tau) y_t}{\sum_{t=1}^{N'} y_t}, \qquad \mathrm{FPR}(\tau) = \frac{\sum_{t=1}^{N'} \hat{y}_t(\tau)(1 - y_t)}{\sum_{t=1}^{N'} (1 - y_t)}.$$

Let $\{\tau_1, \tau_2, \ldots, \tau_K\}$ be a set of thresholds sorted in descending order. The area under the ROC curve is then computed by trapezoidal integration as

$$\text{AUC-ROC} = \sum_{k=1}^{K-1} \big(\mathrm{FPR}(\tau_k + 1) - \mathrm{FPR}(\tau_k)\big) \frac{\mathrm{TPR}(\tau_k) + \mathrm{TPR}(\tau_{k+1})}{2}.$$

AUC-PR. The AUC-PR summarizes the precision–recall trade-off across different thresholds. For a threshold τ, precision and recall are defined as

$$P(\tau) = \frac{\sum_{t=1}^{N'} \hat{y}_t(\tau) y_t}{\sum_{t=1}^{N'} \mathbf{1}(s_t \geq \tau)}, \qquad R(\tau) = \frac{\sum_{t=1}^{N'} \hat{y}_t(\tau) y_t}{\sum_{t=1}^{N'} y_t}.$$

Accordingly, the area under the precision–recall curve is computed as

$$\text{AUC-PR} = \sum_{k=1}^{K-1} \big(R(\tau_k + 1) - R(\tau_k)\big) \frac{P(\tau_k) + P(\tau_{k+1})}{2}.$$

Standard-F1. The standard point-wise F1 score (Std-F1) is defined as the best F1 value achieved over all thresholds:

$$\text{Std-F1} = \max_{\tau} \frac{2P(\tau)R(\tau)}{P(\tau) + R(\tau)}.$$

VUS-ROC. VUS-ROC generalizes ROC-AUC by additionally considering a lag tolerance ℓ around anomalous regions. Instead of evaluating ROC curves only over thresholds τ, it constructs a three-dimensional ROC surface over pairs (τ, ℓ), where $\tau \in \{\tau_1, \ldots, \tau_K\}$ and $\ell \in \{0, \ldots, L\}$. The VUS-ROC is defined as the volume under this surface.

VUS-PR. Similarly, VUS-PR extends the conventional PR-AUC by jointly varying the decision threshold τ and lag tolerance ℓ, thereby forming a precision–recall surface. The VUS-PR is defined as the volume under this surface, offering a threshold-independent and lag-aware extension of standard PR-AUC.

B.3 Implementation Details

We implement TVF-AD in PyTorch. For the offline visual branch, each variable is converted into an RGB time–frequency map using two CWT responses, and visual features are extracted with a frozen CLIP ViT-L/14@336px encoder. For multivariate inputs, variable-wise segment maps are stacked in groups of 4 along the height dimension, resized to 336×336, and converted into variable-aligned patch features, which are further fused by sliding-window mean pooling. In the main setting, we use a window size of 48, train/test/bank strides of 8, segment length 192, CWT parameter $n = 64$, Top-$k = 3$, and post-processing scale factor $\rho = 3$. Unless otherwise specified, the reported results use the offline post-processing protocol described in Section 3.4, which has access to the complete unlabeled test sequence but does not use anomaly labels.

For the temporal branch, we use a 1D CNN encoder with embedding dimension 1024. Non-continuous variables are encoded by a transition-matrix encoder and used to modulate the continuous temporal representation through a FiLM module, after which the temporal and visual features are fused by a gated fusion head. We train the model with AdamW (1×10^{-4}, weight decay 1×10^{-4}) and mixed precision. The training objective is a relational geometry alignment loss, where normalized pairwise cosine distances in the fused representation space are aligned with those of the frozen visual space using SmoothL1 loss. The best checkpoint is selected according to the minimum training loss.

B.4 Details of the Image Construction

This appendix provides the detailed image-construction procedure used in the visual branch. Given a multivariate time series

$$\mathbf{X} = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(D)}] \in \mathbb{R}^{D \times T},$$

we first convert each variable into a full-length time–frequency RGB image and then construct grouped image segments that are fed to the visual encoder.

Per-variable CWT-based RGB map. For the j-th variable $\mathbf{x}^{(j)} = [x_1^{(j)}, \ldots, x_T^{(j)}]$, we construct two time–frequency response maps using discretized continuous wavelet transforms implemented as 1D convolutions over Ω logarithmically spaced frequency bins. Specifically, let the sampling

interval be $dt = 1/f_s$, and let

$$f_\omega \in \left[\frac{1}{n},\, 0.5\right], \qquad \omega = 1, \ldots, \Omega,$$

denote the center frequencies sampled on a logarithmic scale. Following the implementation, the corresponding wavelet scale is defined as

$$s_\omega = \frac{f_c}{f_\omega dt}, \qquad f_c = \frac{w_0}{2\pi},$$

where w_0 is the Morlet central frequency parameter.

For each scale s_ω, we build a Morlet kernel

$$\psi_\omega^{(M)}(\tau) = \cos\left(w_0 \frac{\tau}{s_\omega}\right) \exp\left(-\frac{1}{2}\left(\frac{\tau}{s_\omega}\right)^2\right),$$

and a Ricker kernel

$$\psi_\omega^{(R)}(\tau) = \left(1 - \left(\frac{\tau}{s_\omega}\right)^2\right) \exp\left(-\frac{1}{2}\left(\frac{\tau}{s_\omega}\right)^2\right),$$

where τ indexes the discrete kernel support. Each kernel is L_2-normalized independently. Denoting the resulting kernel banks by

$$\Psi^{(M)}, \Psi^{(R)} \in \mathbb{R}^{\Omega \times 1 \times K},$$

we apply reflect padding followed by 1D convolution to obtain the two response maps:

$$\mathbf{S}_j^{(M)}(t, \omega) = \big(\mathbf{x}^{(j)} * \psi_\omega^{(M)}\big)(t), \qquad \mathbf{S}_j^{(R)}(t, \omega) = \big(\mathbf{x}^{(j)} * \psi_\omega^{(R)}\big)(t),$$

which yields

$$\mathbf{S}_j^{(M)} \in \mathbb{R}^{T \times \Omega}, \qquad \mathbf{S}_j^{(R)} \in \mathbb{R}^{T \times \Omega}.$$

We then take log-amplitudes to obtain the first two image channels:

$$\mathbf{A}_j^{(R)}(t, \omega) = \log\big(1 + |\mathbf{S}_j^{(M)}(t, \omega)|\big), \qquad \mathbf{A}_j^{(G)}(t, \omega) = \log\big(1 + |\mathbf{S}_j^{(R)}(t, \omega)|\big).$$

To avoid train–test leakage, the channel normalization statistics are estimated *only* from the training split. Specifically, for each variable j, we compute

$$a_{j,\min}^{(R)} = \min_{t,\omega} \mathbf{A}_{j,\mathrm{tr}}^{(R)}(t, \omega), \qquad a_{j,\max}^{(R)} = \max_{t,\omega} \mathbf{A}_{j,\mathrm{tr}}^{(R)}(t, \omega),$$

$$a_{j,\min}^{(G)} = \min_{t,\omega} \mathbf{A}_{j,\mathrm{tr}}^{(G)}(t, \omega), \qquad a_{j,\max}^{(G)} = \max_{t,\omega} \mathbf{A}_{j,\mathrm{tr}}^{(G)}(t, \omega),$$

where the subscript "tr" denotes the training split. The normalized red and green channels are then defined as

$$\widetilde{\mathbf{R}}_j(t, \omega) = \mathrm{clip}\left(\frac{\mathbf{A}_j^{(R)}(t, \omega) - a_{j,\min}^{(R)}}{a_{j,\max}^{(R)} - a_{j,\min}^{(R)} + \varepsilon},\, 0,\, 1\right),$$

$$\widetilde{\mathbf{G}}_j(t, \omega) = \mathrm{clip}\left(\frac{\mathbf{A}_j^{(G)}(t, \omega) - a_{j,\min}^{(G)}}{a_{j,\max}^{(G)} - a_{j,\min}^{(G)} + \varepsilon},\, 0,\, 1\right),$$

where $\varepsilon > 0$ is a small constant for numerical stability.

For the third channel, we use normalized log-frequency position. Let f_ω denote the center frequency of the ω-th CWT bin. We define

$$\widetilde{\mathbf{B}}(\omega) = \frac{\log f_\omega - \log f_1}{\log f_\Omega - \log f_1 + \varepsilon}, \qquad \omega = 1, \ldots, \Omega.$$

Since $\widetilde{\mathbf{B}}(\omega)$ depends only on frequency, it is shared across all time steps. The final full-length RGB image of variable j is therefore

$$\mathbf{I}^{(j)}(\omega, t, :) = \Big[\widetilde{\mathbf{R}}_j(t, \omega),\ \widetilde{\mathbf{G}}_j(t, \omega),\ \widetilde{\mathbf{B}}(\omega)\Big] \in [0, 1]^3,$$

which gives

$$\mathbf{I}^{(j)} \in [0,1]^{\Omega \times T \times 3}.$$

Grouping and height-wise composition. After obtaining the full-length image of each variable, we divide the D variables into groups of size d_g, resulting in

$$G = \left\lceil \frac{D}{d_g} \right\rceil$$

groups. For the g-th group, let $\mathcal{G}_g$ denote its variable index set. We concatenate the corresponding variable-wise RGB maps along the height dimension to form a full-length composed image

$$\mathbf{I}^{(g)}_{\text{full}} \in [0,1]^{336 \times T \times 3}.$$

In our implementation, we set

$$\Omega = \frac{336}{d_g},$$

so that after stacking d_g variables vertically, the composed image has a native height of 336. If the last group contains fewer than d_g variables, the remaining slots are padded with zeros.

Temporal segmentation and tail-aligned coverage. Each full-length composed image is partitioned along the temporal axis into segments of length L_{seg}. To maintain exact alignment between temporal regions and patch columns, we require L_{seg} to be divisible by the number of patch columns W_p, and define

$$\delta = \frac{L_{\text{seg}}}{W_p}.$$

Thus, each patch column corresponds to exactly δ original time steps.

We first generate non-overlapping segments:

$$[0, L_{\text{seg}}),\ [L_{\text{seg}}, 2L_{\text{seg}}),\ \ldots$$

If the total length T is not exactly covered, we additionally include one tail-aligned segment ending at

$$t_{\text{end}} = T - (T \bmod \delta),$$

namely

$$[t_{\text{end}} - L_{\text{seg}},\ t_{\text{end}}).$$

This design preserves exact patch-to-time correspondence while improving coverage of the suffix region.

For each temporal segment, we crop the corresponding width range from $\mathbf{I}^{(g)}_{\text{full}}$ and obtain

$$\mathbf{I}^{(g)}_{\text{seg}} \in [0,1]^{336 \times L_{\text{seg}} \times 3}.$$

Each segment is then resized to

$$336 \times 336$$

using bilinear interpolation before being fed into the visual encoder.

Patch-row assignment for each variable. Let the visual encoder tokenize each resized segment into a patch grid with H_p rows and W_p columns. In our implementation with ViT-L/14@336px, we have

$$H_p = W_p = 24.$$

Because the variables within a group are stacked uniformly along the height dimension, each variable occupies exactly

$$\frac{H_p}{d_g}$$

consecutive patch rows. Therefore, for each variable j, its corresponding patch-row set R_j can be identified directly from its vertical position in the composed image.

After this step, the aligned patch-column set C_i, the variable-level visual feature $\mathbf{v}_{i,j}$, and the final window-level visual representation $\mathbf{z}_i^{\text{vis}}$ are computed exactly as described in Section 3.2.

C Sensitivity Analysis

C.1 Numbers of Nearest Neighbors Selected

Table 6: Sensitivity analysis of Top-k on TSB-AD-U and TSB-AD-M.

Dataset	Top-k	VUS-PR	VUS-ROC	AUC-PR	AUC-ROC	Std-F1
TSB-AD-U	1	67.17	94.80	61.00	92.36	64.45
	3	**67.21**	**94.85**	60.81	92.45	64.25
	5	67.17	94.81	61.44	92.54	64.81
	10	67.15	94.78	**62.45**	**92.98**	**65.72**
TSB-AD-M	1	**53.13**	**84.56**	49.00	82.61	51.15
	3	52.93	84.44	48.38	82.43	**51.56**
	5	52.86	84.53	48.65	82.54	51.16
	10	52.66	84.31	**49.10**	**82.86**	51.38

Table 6 shows that TVF-AD is relatively insensitive to the choice of Top-k. While different metrics favor different values of k, $k = 3$ achieves strong and balanced performance on both benchmarks, including the best VUS-PR and VUS-ROC on TSB-AD-U and the best Std-F1 on TSB-AD-M. We therefore use $k = 3$ as the default setting.

C.2 Stride of Time Window

Table 7: Sensitivity analysis of stride on TSB-AD-U and TSB-AD-M.

Dataset	Stride	VUS-PR	VUS-ROC	AUC-PR	AUC-ROC	Std-F1
TSB-AD-U	**8**	**67.21**	**94.85**	60.81	**92.45**	**64.25**
	16	67.17	94.83	**60.92**	92.40	64.17
	24	66.75	94.60	60.63	92.27	63.91
	48	65.14	94.03	58.51	91.64	62.19
TSB-AD-M	**8**	**52.93**	**84.44**	**48.38**	82.43	51.56
	16	52.83	84.30	48.24	82.30	**51.65**
	24	52.89	84.24	47.64	82.42	51.42
	48	51.60	84.21	46.62	**82.52**	50.80

Table 7 presents the sensitivity analysis with respect to the window stride. In all experiments, the image segment length is scaled proportionally with the stride, i.e., it is set to W_p times the stride. Overall, TVF-AD remains stable across moderate stride values, while larger strides generally degrade anomaly-sensitive metrics, particularly VUS-PR and AUC-PR, due to the reduced temporal resolution. Although several ROC-based or threshold-based metrics show small non-monotonic variations, stride $= 8$ provides the strongest and most balanced performance across the two benchmarks. Meanwhile, a smaller stride increases the number of generated images and thus incurs higher computational cost. Considering the trade-off between detection accuracy and efficiency, we adopt stride $= 8$ as the default setting.

C.3 Size of Memory Bank

Table 8 presents the sensitivity analysis of the memory bank ratio. The memory bank is compressed by applying K-means clustering to the training features and retaining only a subset of cluster representatives. Overall, TVF-AD remains relatively stable under moderate memory bank compression, while performance gradually improves as more training features are preserved. The best results on

Table 8: Sensitivity analysis of memory bank ratio on TSB-AD-U and TSB-AD-M.

Dataset	Ratio	VUS-PR	VUS-ROC	AUC-PR	AUC-ROC	Std-F1
TSB-AD-U	0.05	66.83	94.46	60.39	92.00	63.82
	0.10	67.03	94.58	60.61	92.15	64.04
	0.20	67.11	94.69	60.63	92.27	64.08
	0.50	67.20	94.75	60.70	92.34	64.17
	1.00	**67.21**	**94.85**	**60.81**	**92.45**	**64.25**
TSB-AD-M	0.05	52.44	84.23	47.64	82.33	51.08
	0.10	52.67	84.27	47.90	82.39	51.27
	0.20	52.80	84.30	48.04	82.41	51.38
	0.50	52.86	84.32	48.17	82.42	51.43
	1.00	**52.93**	**84.44**	**48.38**	**82.43**	**51.56**

both TSB-AD-U and TSB-AD-M are obtained when the full memory bank is retained. Since the default stride is already set to 8, the number of extracted training features is substantially reduced compared with denser sliding-window settings. Further compressing the memory bank therefore removes additional representative patterns and slightly weakens feature coverage. As shown in Table 8, more aggressive compression leads to a noticeable but still moderate performance degradation. We therefore regard memory bank compression as an optional efficiency-oriented strategy, while retaining the full memory bank for the main experiments.

C.4 Batch Size

Table 9: Sensitivity analysis of batch size on TSB-AD-U and TSB-AD-M.

Dataset	Batch size	VUS-PR	VUS-ROC	AUC-PR	AUC-ROC	Std-F1
TSB-AD-U	16	66.98	94.84	60.83	92.44	64.28
	32	67.08	94.75	60.85	92.44	64.30
	64	67.15	94.78	60.84	92.45	**64.32**
	128	**67.21**	94.65	60.82	**92.46**	64.20
	256	67.19	94.82	**60.87**	92.43	64.20
	512	**67.21**	**94.85**	60.81	92.45	64.25
TSB-AD-M	16	52.48	84.04	48.19	82.09	51.38
	32	52.45	84.12	48.08	82.13	51.42
	64	52.65	84.19	48.18	82.21	51.45
	128	52.69	84.29	48.64	82.31	**51.87**
	256	52.82	84.42	**48.98**	**82.48**	51.81
	512	**52.93**	**84.44**	48.38	82.43	51.56

Table 9 shows that TVF-AD is relatively insensitive to batch size. Performance remains stable across all settings on TSB-AD-U, while larger batch sizes generally perform better on TSB-AD-M. In particular, 512 achieves the best VUS-PR and VUS-ROC, whereas 256 performs best on AUC-PR and AUC-ROC. We therefore use 512 as the default batch size.

C.5 TIME WINDOW SIZE

Table 10: Sensitivity analysis of window size on TSB-AD-U and TSB-AD-M.

Dataset	Window size	VUS-PR	VUS-ROC	AUC-PR	AUC-ROC	Std-F1
TSB-AD-U	24	66.95	**95.04**	60.06	**92.60**	63.41
	32	**67.26**	95.00	60.42	92.55	63.73
	48	67.21	94.85	**60.81**	92.45	**64.25**
	72	66.85	94.84	60.34	92.50	63.94
	192	66.07	94.42	59.99	92.14	63.78
TSB-AD-M	24	51.45	**85.04**	46.15	82.55	49.56
	32	51.63	84.96	46.30	82.52	49.74
	48	**52.93**	84.44	**48.38**	82.43	**51.56**
	72	52.11	84.59	47.75	**82.67**	51.20
	192	49.70	83.01	45.37	81.56	49.78

Table 10 reports the sensitivity analysis of window size on TSB-AD-U and TSB-AD-M. Overall, TVF-AD remains relatively stable across a broad range of window sizes. On TSB-AD-U, smaller windows favor ROC-based metrics, while a window size of 48 achieves the best AUC-PR and Std-F1 and remains competitive on VUS-PR. On TSB-AD-M, a window size of 48 achieves the best VUS-PR, AUC-PR, and Std-F1, whereas other window sizes perform slightly better on the ROC-based metrics. Considering the overall balance across both benchmarks, we use a window size of 48 as the default setting.

C.6 VISUAL BACKBONE

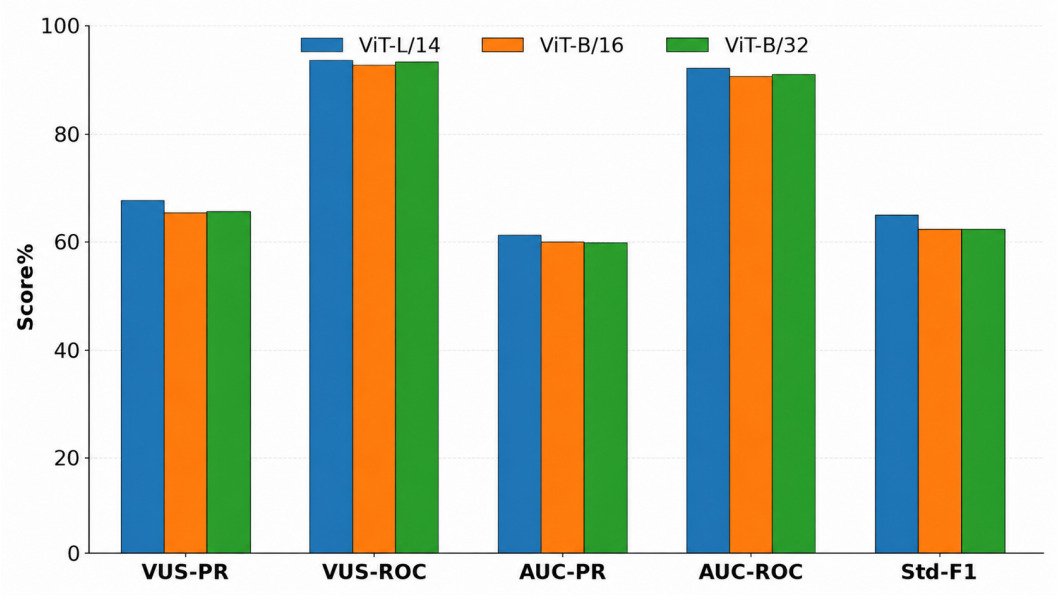


Figure 6: Visual backbone comparison on TSB-AD-U

Figure 6 compares different visual backbones on TSB-AD-U. ViT-L/14 achieves the best performance across all evaluation metrics, indicating that a stronger visual backbone provides more informative representations for anomaly detection. In particular, it achieves a VUS-PR of 67.21, outperforming ViT-B/16 and ViT-B/32 by 1.99 and 1.91 points, respectively.

C.7 Ablation of Training Objectives

Table 11: Comparison of alternative training objectives and component ablations of the proposed geometry-based objective. U and M denote TSB-AD-U and TSB-AD-M, respectively. Higher is better; the best and second-best results are shown in bold and underlined, respectively.

Variant	U VUS-PR	U VUS-ROC	M VUS-PR	M VUS-ROC
Alternative objectives				
Cosine alignment	0.6278	0.9339	0.5044	<u>0.8392</u>
Relational MSE	0.6546	**0.9500**	0.5200	0.8383
Neighbor KL	<u>0.6622</u>	0.9401	0.5175	0.8374
InfoNCE	0.5117	0.9108	0.3920	0.8321
Component ablations of ours				
w/o distance normalization	0.6545	0.9400	<u>0.5203</u>	0.8362
Geometry Alignment (Ours)	**0.6721**	<u>0.9485</u>	**0.5293**	**0.8444**

Table 11 compares alternative training objectives and the effect of relative-distance normalization. Point-wise cosine alignment performs worse than relational objectives, suggesting that preserving inter-sample structure is more effective than directly matching individual fused and visual embeddings. Relational MSE and Neighbor KL remain competitive, while the proposed geometry alignment objective achieves the best VUS-PR on both benchmarks and the best VUS-ROC on TSB-AD-M, while ranking second on VUS-ROC on TSB-AD-U. Removing relative-distance normalization consistently degrades performance, highlighting its importance for stable relational geometry alignment.

C.8 Variable Grouping and Permutation Robustness

We fix the visual grouping size to $d_g = 4$ and vary only the variable grouping in the visual branch, while retaining the original channel order in the temporal branch. Table 12 reports the results on TSB-AD-M.

Table 12: Effect of visual variable grouping on TSB-AD-M with $d_g = 4$. Higher is better; the best and second-best results are shown in bold and underlined, respectively.

Grouping strategy	VUS-PR	VUS-ROC	AUC-PR	AUC-ROC	Std-F1
Original order	0.5293	<u>0.8444</u>	0.4838	**0.8243**	<u>0.5156</u>
Random permutation	0.5288	0.8431	<u>0.4898</u>	0.8197	0.5138
Correlation grouping	**0.5357**	0.8438	**0.4900**	0.8214	**0.5169**
Correlation groups + random inner order	<u>0.5318</u>	**0.8447**	<u>0.4898</u>	<u>0.8220</u>	0.5130

For correlation-based grouping, we compute the absolute Pearson correlation between variables using only the normal training segment and define the pairwise distance as $d_{ij} = 1 - |\operatorname{corr}(x_i, x_j)|$. We then apply average-linkage hierarchical clustering with optimal leaf ordering and partition consecutive variables into groups of four. Neither test data nor anomaly labels are used to construct the groups.

Overall, the performance varies only mildly across different grouping strategies. Correlation grouping achieves the best VUS-PR, AUC-PR, and Std-F1, while correlation grouping with randomized within-group order achieves the best VUS-ROC. The original variable order performs best on AUC-ROC and remains competitive on the other metrics. Random permutation and within-group randomization lead to only small performance changes. These results suggest that correlation-based grouping can slightly improve anomaly-sensitive metrics, while TVF-AD remains generally robust to the specific variable grouping and ordering used for visual construction.

C.9 Quantitative Analysis of Temporal–Visual Complementarity

We evaluate the temporal-only, visual-only, and full models on each test series and compare their per-series VUS-PR scores. Table 13 reports the number of series on which each variant achieves the highest score.

Table 13: Number of series on which each variant achieves the highest VUS-PR. Bold indicates the largest count in each row.

Dataset	Temporal-only	Visual-only	Full TVF-AD
TSB-AD-U	98	108	**144**
TSB-AD-M	60	47	**73**
Total	158	155	**217**

The full model achieves the highest VUS-PR on 217 of 530 series, exceeding the temporal-only and visual-only variants by 59 and 62 wins, respectively. Both individual branches also perform best on substantial subsets, supporting their complementary strengths.

Learned gate behavior. We further analyze the learned vector gate on TSB-AD-M. We average gate values over feature dimensions to obtain a window-level gate value and further average over windows within each series to obtain a series-level value. Using 0.5 as the boundary, we categorize windows and series as temporal-dominant or visual-dominant. Larger gate values indicate greater weight on the temporal branch.

Table 14: Temporal and visual dominance of the learned gate on TSB-AD-M. Series-level percentages weight each series equally, whereas window-level percentages are computed over all windows.

	Series-level dominance		Window-level dominance	
Dataset	**Temporal**	**Visual**	**Temporal**	**Visual**
TSB-AD-M	106/180 (58.89%)	74/180 (41.11%)	31.31%	68.69%

The difference between series-level and window-level statistics reflects unequal series lengths: several long visual-dominant series, particularly in the Medical domain, contribute many windows. Gate preferences also vary across domains, with Medical and Human Activity favoring the visual branch, and Facility, Sensor, Environment, and Finance favoring the temporal branch. These observations suggest that the gate varies across inputs rather than uniformly favoring one modality.

C.10 Sensitivity to Window-to-Point Score Aggregation

We examine the effect of window-to-point score aggregation by comparing arithmetic mean, harmonic mean, median, and center-weighted mean. The trained model, window-level anomaly scores, memory bank, retrieval procedure, and evaluation protocol are kept fixed; only the aggregation operator is changed.

Performance remains generally stable across the four aggregation operators, with harmonic-mean aggregation providing the strongest overall performance. On TSB-AD-U, it achieves the best VUS-ROC, AUC-PR, AUC-ROC, and Std-F1, while Arithmetic mean attains a slightly higher VUS-PR. On TSB-AD-M, harmonic mean achieves the best VUS-PR, AUC-PR, and Std-F1, whereas Median and Arithmetic mean perform better on VUS-ROC and AUC-ROC, respectively. Overall, the small differences across aggregation strategies indicate that TVF-AD is not highly sensitive to this downstream choice, while harmonic mean provides the most consistent performance across the two benchmarks.

Table 15: Sensitivity to window-to-point score aggregation. Higher is better; the best and second-best results within each benchmark are shown in bold and underlined, respectively.

Aggregation	Dataset	VUS-PR	VUS-ROC	AUC-PR	AUC-ROC	Std-F1
Arithmetic mean	TSB-AD-U	**0.6730**	0.9402	<u>0.6075</u>	<u>0.9160</u>	<u>0.6412</u>
Harmonic mean	TSB-AD-U	<u>0.6721</u>	**0.9485**	**0.6081**	**0.9245**	**0.6425**
Median	TSB-AD-U	0.6719	<u>0.9422</u>	0.6047	0.9156	0.6388
Center-weighted mean	TSB-AD-U	0.6715	0.9405	0.6064	<u>0.9160</u>	0.6401
Arithmetic mean	TSB-AD-M	<u>0.5237</u>	<u>0.8493</u>	<u>0.4756</u>	**0.8276**	<u>0.5089</u>
Harmonic mean	TSB-AD-M	**0.5293**	0.8444	**0.4838**	0.8243	**0.5156**
Median	TSB-AD-M	0.5230	**0.8498**	0.4731	<u>0.8274</u>	0.5085
Center-weighted mean	TSB-AD-M	0.5224	0.8492	0.4710	0.8264	0.5057

C.11 Comparison of Alternative Fusion Mechanisms

We compare single-branch variants and alternative fusion mechanisms. Let T and V denote the temporal and visual representations, respectively. For scalar gating, $g \in (0, 1)$ is a learned scalar; for vector gating, $g \in (0, 1)^d$ provides a separate weight for each feature dimension. Here, $[T; V]$ denotes concatenation and $\odot$ denotes element-wise multiplication.

Table 16: Comparison of alternative fusion mechanisms on TSB-AD-U and TSB-AD-M. Higher is better; the best and second-best results within each benchmark are shown in bold and underlined, respectively.

Dataset	Fusion	Rule	VUS-PR	VUS-ROC	AUC-PR	AUC-ROC	Std-F1
TSB-AD-U	Temporal only	$z = T$	0.5903	0.9221	0.5376	0.8928	0.5818
TSB-AD-U	Visual only	$z = V$	0.4933	0.8975	0.3881	0.8663	0.4319
TSB-AD-U	Equal average	$z = (T + V)/2$	0.6560	0.9407	0.5931	0.9165	<u>0.6354</u>
TSB-AD-U	Concatenation + MLP	$z = \mathrm{MLP}([T; V])$	<u>0.6676</u>	**0.9495**	<u>0.5988</u>	**0.9252**	0.6317
TSB-AD-U	Scalar gate	$z = gT + (1 - g)V$	0.6442	0.9468	0.5745	0.9224	0.6065
TSB-AD-U	Vector gate	$z = g \odot T + (1 - g) \odot V$	**0.6721**	<u>0.9485</u>	**0.6081**	<u>0.9245</u>	**0.6425**
TSB-AD-M	Temporal only	$z = T$	0.4554	0.8287	0.4131	0.8044	0.4657
TSB-AD-M	Visual only	$z = V$	0.4543	<u>0.8613</u>	0.3872	<u>0.8307</u>	0.4417
TSB-AD-M	Equal average	$z = (T + V)/2$	0.5144	0.8490	<u>0.4821</u>	0.8303	<u>0.5091</u>
TSB-AD-M	Concatenation + MLP	$z = \mathrm{MLP}([T; V])$	<u>0.5232</u>	0.8520	0.4673	0.8289	0.5074
TSB-AD-M	Scalar gate	$z = gT + (1 - g)V$	0.5018	**0.8637**	0.4373	**0.8372**	0.4879
TSB-AD-M	Vector gate	$z = g \odot T + (1 - g) \odot V$	**0.5293**	0.8444	**0.4838**	0.8243	**0.5156**

Vector gating achieves the best VUS-PR, AUC-PR, and Std-F1 on both benchmarks, demonstrating strong overall performance across ranking- and threshold-based metrics. On TSB-AD-U, concatenation with an MLP attains slightly higher VUS-ROC and AUC-ROC, while on TSB-AD-M scalar gating performs best on the two ROC-based metrics. Equal averaging also remains competitive, but is generally outperformed by vector gating. Overall, these results support vector gating as the default fusion mechanism due to its consistently strong performance across both benchmarks.

C.12 Ablation of Temporal-Branch Conditioning

We further investigate the role of discrete-state conditioning in the temporal branch on TSB-AD-M. In the default design, non-continuous variables are encoded by the state-transition encoder and used to modulate the continuous temporal representation through FiLM. Specifically, the discrete representation generates feature-wise scaling and shifting parameters for the continuous temporal features. We compare this design with three alternatives: removing discrete conditioning entirely, concatenating the continuous and discrete representations followed by an MLP, and additive conditioning.

Table 17: Ablation of temporal-branch conditioning on TSB-AD-M. The best and second-best results for each metric are shown in **bold** and underlined, respectively.

Temporal conditioning	VUS-PR	VUS-ROC	AUC-PR	AUC-ROC	Std-F1
FiLM-based discrete conditioning	**0.5293**	**0.8444**	**0.4838**	0.8243	**0.5156**
No discrete conditioning	0.5268	0.8407	0.4805	**0.8304**	0.5135
Concatenation + MLP	0.5236	0.8424	0.4787	0.8220	0.5125
Additive conditioning	0.5219	0.8408	0.4747	0.8292	0.5100

As shown in Table 17, FiLM-based discrete conditioning achieves the best performance on VUS-PR, VUS-ROC, AUC-PR, and Std-F1. In particular, it improves VUS-PR from 0.5268 without discrete conditioning to 0.5293, while increasing VUS-ROC from 0.8407 to 0.8444. It also outperforms concatenation-based and additive conditioning on both VUS metrics, indicating that feature-wise modulation is effective for incorporating discrete-state information.

Removing discrete conditioning remains competitive and achieves the highest AUC-ROC (0.8304), compared with 0.8243 for FiLM-based conditioning. Overall, FiLM provides the strongest performance across most metrics, suggesting that discrete operating states offer useful context for adapting continuous temporal representations.

C.13 Ablation of Postprocessing Components

Rationale for post-processing. The base anomaly score of TVF-AD is derived from fixed-length window representations and therefore mainly captures anomalies at a single temporal scale. To handle anomalies with different durations, we introduce short- and long-scale raw-pattern matching, while a local innovation signal captures abrupt point-wise deviations that may be diluted by window-level modeling. Together, these signals complement the learned representation with multi-scale and local anomaly cues. The post-processing hyperparameters α, β, ρ, and p are selected on the official TSB-AD tuning set and fixed for all evaluation experiments, without tuning on the evaluation set.

Table 18: Ablation of the three auxiliary post-processing signals on TSB-AD-U/M. The best and second-best results for each dataset and metric are shown in **bold** and underlined, respectively.

Variant	Dataset	VUS-PR	VUS-ROC	AUC-PR	AUC-ROC
Base only	U	0.5802	0.9236	0.5072	0.8928
	M	0.5061	0.8482	0.4572	0.8250
w/o short-scale pattern	U	0.6461	0.9373	0.5694	0.9103
	M	0.5169	0.8492	0.4567	0.8260
w/o long-scale pattern	U	0.6044	0.9387	0.5449	0.9103
	M	0.4630	0.8480	0.4138	0.8211
w/o local innovation	U	0.6115	0.9387	0.5487	0.9138
	M	0.5227	**0.8536**	0.4759	**0.8309**
Full	U	**0.6721**	**0.9485**	**0.6081**	**0.9245**
	M	**0.5293**	0.8444	**0.4838**	0.8243

Table 18 shows that the auxiliary post-processing signals provide complementary information to the learned anomaly score. The full configuration performs best across all metrics on TSB-AD-U and achieves the best VUS-PR and AUC-PR on TSB-AD-M. Removing the long-scale pattern cue causes the largest VUS-PR degradation on both benchmarks, while removing local innovation slightly improves the ROC-based metrics on TSB-AD-M. Overall, the results support the joint use of multi-scale raw-pattern matching and local innovation.

To assess whether the gains of TVF-AD mainly come from post-processing, we apply the same post-processing procedure with identical hyperparameters to PaAno and CARLA on TSB-AD-M. As shown in Table 19, post-processing consistently improves both baselines, indicating that the multi-scale raw-pattern and local-innovation cues provide generally useful complementary information.

Table 19: Effect of applying the same post-processing (PP) to alternative detectors on TSB-AD-M.

Method	VUS-PR	VUS-ROC	AUC-PR	AUC-ROC
PaAno	0.4267	0.7918	0.3815	0.7604
PaAno + PP	0.4746	0.8131	0.4213	0.7938
CARLA	0.3296	0.6849	0.2809	0.6612
CARLA + PP	0.4347	0.8079	0.3957	0.7877
TVF-AD + PP	**0.5293**	**0.8444**	**0.4838**	**0.8243**

Nevertheless, TVF-AD still outperforms both methods across all reported metrics under the same post-processing protocol. This suggests that its performance advantage cannot be attributed solely to post-processing and further supports the effectiveness of temporal–visual representation learning.

C.14 Ablation of Visual Prior

Table 20: Comparison of different visual representations and encoders on TSB-AD-U. The full TVF-AD results are taken from the main experiment.

ID	Setting	VUS-PR	VUS-ROC	AUC-PR	AUC-ROC	Std-F1
S0	Temporal-Only	0.5903	0.9221	0.5376	0.8928	0.5818
S1	Compact CWT vector + MLP	0.3072	0.7738	0.2642	0.7240	0.3166
S2	CWT image + frozen random ViT	0.4825	0.8826	0.3777	0.8618	0.4306
S3	CWT image + MLP	0.4783	0.8226	0.3831	0.8113	0.4318
S4	CWT image + frozen random Transformer1D	0.4627	0.8374	0.3762	0.8068	0.4380
S5	CWT image + pretrained ViT (Visual-Only)	0.4933	0.8975	0.3881	0.8663	0.4319
Full	Full TVF-AD	**0.6721**	**0.9485**	**0.6081**	**0.9245**	**0.6425**

Effect of visual representation and pretrained visual priors. Table 20 further disentangles the contributions of frequency information, image-based representation, visual encoder architecture, pretrained visual priors, and temporal–visual fusion on TSB-AD-U.

The compact CWT representation (S1) performs substantially worse than the image-based variants, indicating that frequency information alone is insufficient to explain the gains. In particular, replacing the compact CWT vector with structured CWT image inputs while using an MLP encoder (S3) improves VUS-PR from 0.3072 to 0.4783 and AUC-PR from 0.2642 to 0.3831, suggesting that preserving the structured time–frequency representation provides substantial benefit.

Among randomly initialized encoders, the image-oriented ViT (S2) achieves higher VUS-PR, VUS-ROC, AUC-PR, and AUC-ROC than the row-wise Transformer1D (S4), supporting the benefit of explicitly modeling the two-dimensional organization of the CWT image. Replacing the random ViT with a pretrained visual encoder (S5) further improves all five metrics, increasing VUS-PR from 0.4825 to 0.4933 and AUC-PR from 0.3777 to 0.3881. This result demonstrates that pretrained visual priors provide additional useful information beyond the image-oriented architecture itself.

Finally, combining the pretrained visual representation with the temporal branch through adaptive fusion yields the full TVF-AD model, which achieves the best performance across all five metrics, including a VUS-PR of 0.6721, an AUC-PR of 0.6081, and a Std-F1 of 0.6425. Compared with the visual-only variant S5, the full model improves VUS-PR by 0.1788 and AUC-PR by 0.2200, demonstrating the substantial benefit of incorporating complementary temporal information. Overall, these results show that the gains of TVF-AD cannot be attributed to frequency information alone; rather, structured image representation, image-oriented modeling, pretrained visual priors, and temporal–visual fusion jointly contribute to the final performance.

D Detailed Experimental Results

D.1 Visualization

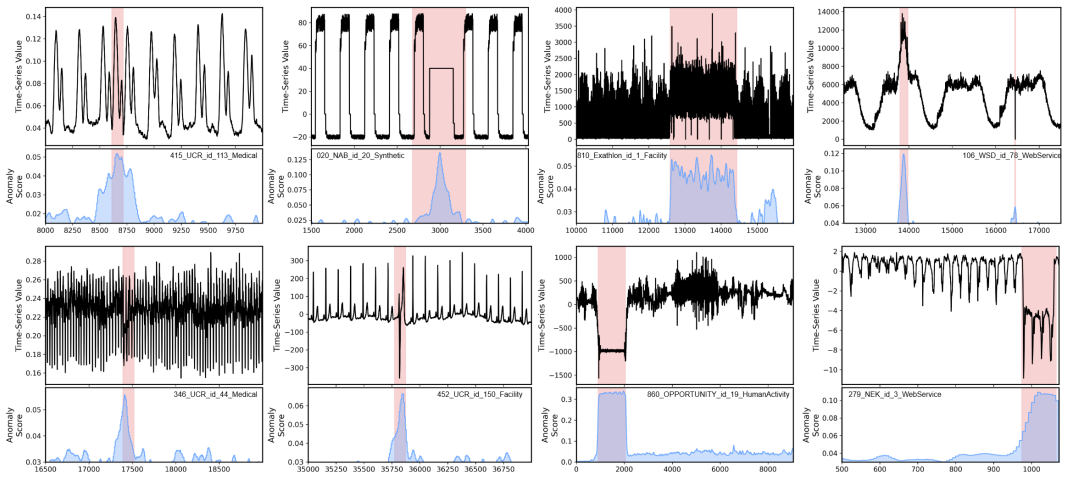


Figure 7: Test results on selected time series. The shaded regions indicate ground-truth anomalies, and the blue curves denote the anomaly scores produced by TVF-AD.

Figure 7 presents several representative test examples. As can be seen, TVF-AD assigns consistently high anomaly scores to the ground-truth anomalous regions across diverse anomaly patterns, while keeping relatively low responses on normal segments. This result suggests that TVF-AD can effectively capture various types of anomalies, including point-wise, contextual, shape-related, seasonal, collective, and trend anomalies.

D.2 Detailed results by subset.

Tables 21 and 22 present the primary and supplementary metrics on TSB-AD-U, respectively. Tables 23 and 24 present the corresponding results on TSB-AD-M. All results are reported as mean $\pm$ standard deviation over 5 random seeds. Besides the primary evaluation metrics, we additionally report PA-F1, Event-based-F1, and Affiliation-F as supplementary reference metrics. TVF-AD shows relatively lower performance on several subsets, such as GHL, OPPORTUNITY, and CreditCard. This may be partly attributed to their high-dimensional multivariate structure and sparse-channel anomalies, where anomalous behavior affects only one or a few variables while most variables remain normal. In such cases, the anomaly signal can be diluted when information is aggregated across variables, making these localized anomalies more difficult to distinguish.

Table 21: Detailed results by subset on TSB-AD-U: primary metrics.

Dataset	Subset	#Series	AUC-PR	AUC-ROC	VUS-PR	VUS-ROC	Standard-F1
TSB-AD-U	CATSv2	1	0.5273 ± 0.0006	0.7492 ± 0.0015	0.3922 ± 0.0042	0.7617 ± 0.0020	0.5871 ± 0.0025
	Daphnet	1	0.4856 ± 0.0054	0.9486 ± 0.0015	0.5072 ± 0.0058	0.9534 ± 0.0014	0.5177 ± 0.0083
	Exathlon	30	0.9784 ± 0.0005	0.9984 ± 0.0001	0.9802 ± 0.0004	0.9987 ± 0.0001	0.9752 ± 0.0001
	IOPS	15	0.4863 ± 0.0037	0.9372 ± 0.0050	0.5581 ± 0.0077	0.9768 ± 0.0010	0.5378 ± 0.0036
	LTDB	8	0.6835 ± 0.0007	0.8661 ± 0.0007	0.7489 ± 0.0008	0.9028 ± 0.0005	0.6461 ± 0.0012
	MGAB	8	0.1814 ± 0.0053	0.9388 ± 0.0015	0.1677 ± 0.0055	0.9525 ± 0.0015	0.3343 ± 0.0116
	MITDB	7	0.4105 ± 0.0018	0.8936 ± 0.0007	0.4365 ± 0.0019	0.9226 ± 0.0004	0.4865 ± 0.0015
	MSL	7	0.4113 ± 0.0053	0.8272 ± 0.0026	0.4909 ± 0.0050	0.8480 ± 0.0047	0.5239 ± 0.0085
	NAB	23	0.5627 ± 0.0056	0.8441 ± 0.0045	0.5986 ± 0.0064	0.8692 ± 0.0035	0.5749 ± 0.0034
	NEK	8	0.8133 ± 0.0050	0.9780 ± 0.0025	0.8666 ± 0.0089	0.9874 ± 0.0025	0.8367 ± 0.0060
	OPPORTUNITY	27	0.6535 ± 0.0126	0.9284 ± 0.0018	0.6704 ± 0.0125	0.9331 ± 0.0019	0.7362 ± 0.0073
	Power	1	0.2640 ± 0.0036	0.7078 ± 0.0030	0.2645 ± 0.0039	0.7283 ± 0.0028	0.2563 ± 0.0025
	SED	2	0.8565 ± 0.0049	0.9958 ± 0.0002	0.9805 ± 0.0009	0.9994 ± 0.0000	0.8580 ± 0.0071
	SMAP	17	0.6932 ± 0.0008	0.9293 ± 0.0021	0.7208 ± 0.0006	0.9386 ± 0.0030	0.7295 ± 0.0011
	SMD	33	0.7957 ± 0.0029	0.9800 ± 0.0003	0.8124 ± 0.0028	0.9830 ± 0.0002	0.7755 ± 0.0040
	SVDB	18	0.7033 ± 0.0008	0.9793 ± 0.0002	0.7532 ± 0.0005	0.9852 ± 0.0003	0.6989 ± 0.0012
	SWaT	1	0.7461 ± 0.0013	0.8363 ± 0.0019	0.3954 ± 0.0072	0.7148 ± 0.0044	0.7908 ± 0.0005
	Stock	8	0.1251 ± 0.0007	0.6368 ± 0.0015	0.8136 ± 0.0012	0.9092 ± 0.0005	0.1958 ± 0.0012
	TAO	2	0.1728 ± 0.0012	0.6232 ± 0.0019	0.9386 ± 0.0006	0.9610 ± 0.0003	0.2561 ± 0.0007
	TODS	13	0.2586 ± 0.0027	0.7585 ± 0.0030	0.7520 ± 0.0032	0.9198 ± 0.0010	0.3092 ± 0.0041
	UCR	70	0.4271 ± 0.0103	0.9181 ± 0.0009	0.4673 ± 0.0038	0.9317 ± 0.0014	0.4885 ± 0.0072
	WSD	20	0.6176 ± 0.0100	0.9903 ± 0.0005	0.6308 ± 0.0076	0.9953 ± 0.0006	0.6261 ± 0.0089
	YAHOO	30	0.8341 ± 0.0017	0.9829 ± 0.0019	0.8395 ± 0.0048	0.9825 ± 0.0018	0.8532 ± 0.0040
TSB-AD-U Average			0.6081 ± 0.0028	0.9245 ± 0.0005	0.6721 ± 0.0014	0.9485 ± 0.0004	0.6425 ± 0.0027

Table 22: Detailed results by subset on TSB-AD-U: supplementary metrics.

Dataset	Subset	#Series	PA-F1	Event-based-F1	R-based-F1	Affiliation-F
TSB-AD-U	CATSv2	1	0.6350 ± 0.0162	0.5035 ± 0.0085	0.1116 ± 0.0087	0.7240 ± 0.0029
	Daphnet	1	0.9950 ± 0.0000	0.7888 ± 0.0100	0.4108 ± 0.0129	0.9566 ± 0.0049
	Exathlon	30	0.9933 ± 0.0000	0.9933 ± 0.0000	0.7052 ± 0.0064	0.9962 ± 0.0001
	IOPS	15	0.7910 ± 0.0043	0.8367 ± 0.0005	0.5042 ± 0.0013	0.9273 ± 0.0016
	LTDB	8	0.8870 ± 0.0006	0.8174 ± 0.0009	0.5867 ± 0.0012	0.8987 ± 0.0012
	MGAB	8	0.1977 ± 0.0071	0.1857 ± 0.0043	0.0827 ± 0.0042	0.7736 ± 0.0019
	MITDB	7	0.7790 ± 0.0014	0.5986 ± 0.0020	0.4115 ± 0.0015	0.8788 ± 0.0010
	MSL	7	0.8622 ± 0.0046	0.6595 ± 0.0031	0.4644 ± 0.0157	0.8861 ± 0.0037
	NAB	23	0.9509 ± 0.0054	0.9010 ± 0.0102	0.5114 ± 0.0136	0.9474 ± 0.0036
	NEK	8	0.9512 ± 0.0107	0.9475 ± 0.0061	0.5379 ± 0.0129	0.9725 ± 0.0031
	OPPORTUNITY	27	0.8959 ± 0.0032	0.7852 ± 0.0084	0.3317 ± 0.0107	0.9325 ± 0.0022
	Power	1	0.9398 ± 0.0016	0.8571 ± 0.0000	0.3630 ± 0.0000	0.8748 ± 0.0021
	SED	2	0.8425 ± 0.0021	0.8386 ± 0.0028	0.7598 ± 0.0085	0.9861 ± 0.0004
	SMAP	17	0.9428 ± 0.0004	0.9405 ± 0.0000	0.6225 ± 0.0051	0.9833 ± 0.0000
	SMD	33	0.9242 ± 0.0013	0.8757 ± 0.0021	0.4464 ± 0.0100	0.9754 ± 0.0005
	SVDB	18	0.8316 ± 0.0022	0.7724 ± 0.0027	0.5541 ± 0.0043	0.9356 ± 0.0012
	SWaT	1	0.8523 ± 0.0002	0.4588 ± 0.0096	0.2152 ± 0.0246	0.7013 ± 0.0033
	Stock	8	0.1960 ± 0.0011	0.1936 ± 0.0012	0.1317 ± 0.0024	0.6933 ± 0.0005
	TAO	2	0.2566 ± 0.0003	0.2528 ± 0.0005	0.1608 ± 0.0072	0.6831 ± 0.0004
	TODS	13	0.4758 ± 0.0044	0.4087 ± 0.0022	0.3959 ± 0.0071	0.7725 ± 0.0035
	UCR	70	0.6371 ± 0.0050	0.5367 ± 0.0038	0.3065 ± 0.0038	0.9333 ± 0.0008
	WSD	20	0.8656 ± 0.0026	0.8702 ± 0.0037	0.5119 ± 0.0074	0.9694 ± 0.0006
	YAHOO	30	0.8568 ± 0.0013	0.8418 ± 0.0015	0.8158 ± 0.0017	0.9747 ± 0.0013
TSB-AD-U Average			0.7942 ± 0.0014	0.7410 ± 0.0013	0.4738 ± 0.0025	0.9319 ± 0.0002

Table 23: Detailed results by subset on TSB-AD-M: primary metrics.

Dataset	Subset	#Series	AUC-PR	AUC-ROC	VUS-PR	VUS-ROC	Standard-F1
TSB-AD-M	CATSv2	5	0.1765 ± 0.0121	0.6778 ± 0.0043	0.1223 ± 0.0049	0.6423 ± 0.0040	0.2366 ± 0.0106
	CreditCard	1	0.0364 ± 0.0033	0.7553 ± 0.0029	0.0871 ± 0.0054	0.7415 ± 0.0030	0.1028 ± 0.0010
	Daphnet	1	0.3603 ± 0.0180	0.9376 ± 0.0044	0.3782 ± 0.0224	0.9416 ± 0.0046	0.4704 ± 0.0198
	Exathlon	25	0.9955 ± 0.0007	0.9993 ± 0.0001	0.9963 ± 0.0005	0.9996 ± 0.0001	0.9840 ± 0.0007
	GECCO	1	0.2518 ± 0.0074	0.7141 ± 0.0011	0.0449 ± 0.0008	0.5467 ± 0.0008	0.3760 ± 0.0096
	GHL	23	0.0149 ± 0.0007	0.5259 ± 0.0053	0.0161 ± 0.0008	0.5286 ± 0.0052	0.0465 ± 0.0009
	Genesis	1	0.9003 ± 0.0843	0.9997 ± 0.0002	0.9638 ± 0.0352	0.9999 ± 0.0001	0.8942 ± 0.0389
	LTDB	4	0.5903 ± 0.0072	0.9035 ± 0.0029	0.6337 ± 0.0101	0.9274 ± 0.0035	0.5563 ± 0.0059
	MITDB	11	0.4290 ± 0.0066	0.8910 ± 0.0010	0.3476 ± 0.0030	0.9054 ± 0.0009	0.4560 ± 0.0038
	MSL	14	0.5697 ± 0.0196	0.8965 ± 0.0083	0.6538 ± 0.0326	0.9222 ± 0.0077	0.6191 ± 0.0122
	OPPORTUNITY	7	0.1909 ± 0.0473	0.5105 ± 0.0299	0.1909 ± 0.0255	0.4751 ± 0.0169	0.2975 ± 0.0563
	PSM	1	0.1889 ± 0.0083	0.7126 ± 0.0075	0.1979 ± 0.0046	0.6586 ± 0.0046	0.3052 ± 0.0052
	SMAP	25	0.6123 ± 0.0154	0.9182 ± 0.0052	0.6365 ± 0.0187	0.9239 ± 0.0043	0.6477 ± 0.0142
	SMD	20	0.4610 ± 0.0205	0.8809 ± 0.0051	0.4144 ± 0.0101	0.8666 ± 0.0037	0.5035 ± 0.0147
	SVDB	28	0.5725 ± 0.0031	0.9107 ± 0.0017	0.6039 ± 0.0037	0.9345 ± 0.0019	0.5787 ± 0.0026
	SWaT	2	0.5377 ± 0.0025	0.8415 ± 0.0021	0.3734 ± 0.0069	0.7097 ± 0.0052	0.6605 ± 0.0028
	TAO	11	0.1097 ± 0.0010	0.5907 ± 0.0025	0.8186 ± 0.0023	0.8939 ± 0.0016	0.1841 ± 0.0012
TSB-AD-M Average			0.4838 ± 0.0043	0.8243 ± 0.0017	0.5293 ± 0.0042	0.8444 ± 0.0014	0.5156 ± 0.0043

Table 24: Detailed results by subset on TSB-AD-M: supplementary metrics.

Dataset	Subset	#Series	PA-F1	Event-based-F1	R-based-F1	Affiliation-F
TSB-AD-M	CATSv2	5	0.4684 ± 0.0354	0.3063 ± 0.0162	0.4791 ± 0.0233	0.7541 ± 0.0056
	CreditCard	1	0.0515 ± 0.0043	0.0500 ± 0.0043	0.0315 ± 0.0033	0.6821 ± 0.0016
	Daphnet	1	0.8178 ± 0.0379	0.5701 ± 0.0253	0.3781 ± 0.0292	0.9138 ± 0.0089
	Exathlon	25	1.0000 ± 0.0001	0.9999 ± 0.0001	0.9164 ± 0.0178	0.9993 ± 0.0000
	GECCO	1	0.4096 ± 0.0074	0.1260 ± 0.0011	0.4556 ± 0.0498	0.7701 ± 0.0039
	GHL	23	0.0936 ± 0.0090	0.0472 ± 0.0018	0.1607 ± 0.0101	0.8063 ± 0.0011
	Genesis	1	0.5549 ± 0.0026	0.5549 ± 0.0026	0.3161 ± 0.0442	0.9568 ± 0.0120
	LTDB	4	0.7438 ± 0.0081	0.6979 ± 0.0077	0.5188 ± 0.0091	0.9121 ± 0.0022
	MITDB	11	0.6467 ± 0.0026	0.5251 ± 0.0023	0.3124 ± 0.0042	0.8941 ± 0.0029
	MSL	14	0.7746 ± 0.0106	0.7231 ± 0.0231	0.5962 ± 0.0167	0.9542 ± 0.0114
	OPPORTUNITY	7	0.4116 ± 0.0591	0.3269 ± 0.0536	0.4724 ± 0.0249	0.7682 ± 0.0189
	PSM	1	0.6423 ± 0.0177	0.3082 ± 0.0042	0.4374 ± 0.0130	0.8165 ± 0.0045
	SMAP	25	0.8175 ± 0.0121	0.6959 ± 0.0152	0.5543 ± 0.0238	0.9328 ± 0.0011
	SMD	20	0.6815 ± 0.0170	0.4943 ± 0.0040	0.4389 ± 0.0091	0.8495 ± 0.0029
	SVDB	28	0.7476 ± 0.0060	0.6640 ± 0.0049	0.4777 ± 0.0044	0.8796 ± 0.0018
	SWaT	2	0.7436 ± 0.0161	0.4693 ± 0.0064	0.6172 ± 0.0538	0.7585 ± 0.0013
	TAO	11	0.1837 ± 0.0011	0.1824 ± 0.0011	0.1645 ± 0.0031	0.6806 ± 0.0002
TSB-AD-M Average			0.6350 ± 0.0051	0.5501 ± 0.0052	0.4824 ± 0.0071	0.8756 ± 0.0007

D.3 Efficiency Analysis

To evaluate the practical efficiency of different methods, we report the average run time including both training and inference, averaged across the time series within each benchmark. For all baseline methods other than PaAno, CARLA, and TVF-AD, the reported run times are directly taken from the TSB-AD benchmark (Liu & Paparrizos, 2024). In that benchmark, statistical and conventional machine learning methods were executed on an AMD EPYC 7713 CPU, whereas neural network-based and Transformer/foundation-model-based methods were evaluated with single-GPU acceleration; more generally, TSB-AD states that runtime measurement is conducted by default on a single CPU process, with a single GPU used to accelerate neural and foundation-model methods. For a fairer comparison with our implementations, we re-measured the run times of PaAno, CARLA, and TVF-AD under the same evaluation protocol and on the same hardware platform, using an Intel Core i7-13700 CPU, 32 GB RAM, and a single NVIDIA RTX 4090 GPU. The reported average run time includes both training and inference, but excludes the time required for metric computation. For reference, under the default setting, one complete run of TVF-AD, excluding metric computation, takes approximately 64 minutes on TSB-AD-U and 113 minutes on TSB-AD-M.

Table 25: Runtime breakdown of TVF-AD on TSB-AD-U and TSB-AD-M. Visual feature extraction includes CWT-based image construction and frozen visual-encoder inference. Only relative runtime percentages are reported.

Stage	TSB-AD-U	TSB-AD-M	Overall
Visual Feature Extraction	**52.22%**	**71.05%**	**62.77%**
Training & Base Scoring	28.19%	19.84%	23.51%
Startup / Shutdown	12.54%	5.55%	8.62%
Auxiliary Scoring	2.88%	1.68%	2.21%
Fusion & Output	0.56%	0.47%	0.51%
Other Synchronization / Logging	3.62%	1.41%	2.38%
Total	100.00%	100.00%	100.00%

Table 25 further decomposes the computational cost of TVF-AD. Since this stage-wise profiling was conducted on a different GPU from the overall runtime comparison above, we report only the relative runtime percentages rather than absolute times. Visual feature extraction, which includes both CWT-based image construction and frozen visual-encoder inference, is the dominant computational component, accounting for 52.22% of the runtime on TSB-AD-U and 71.05% on TSB-AD-M. In contrast, model training and base anomaly scoring account for 28.19% and 19.84%, respectively, while the remaining stages contribute only a relatively small fraction of the total cost.

The larger visual-processing proportion on TSB-AD-M is mainly due to the multivariate setting: each series contains more variables, requiring more variable-wise CWT maps and consequently more grouped visual inputs to be processed by the frozen vision encoder. This observation indicates that the main computational bottleneck of TVF-AD lies in offline visual feature construction rather than in the lightweight temporal encoder, fusion module, or downstream scoring procedure. Importantly, visual features are extracted offline and can be reused during subsequent training and inference stages, which limits the repeated computational overhead of the visual backbone.

Notably, the proposed fusion and auxiliary scoring stages together account for less than 3% of the overall runtime, suggesting that the performance gains of TVF-AD introduce only limited additional overhead beyond visual feature construction.